DIGITAL DESIGN AND COMPUTER ORGANIZATION LABORATORY

UE24CS251A

Computer Science and Engineering

Questions

1. Design and implement 2 bit up synchronous counter using JK flipflop
2. Design and implement 2 bit down synchronous counter using JK flipflop
3. Design and implement 2 bit up down synchronous counter using JK flipflop
4. Design 2 bit ripple up counter using D flipflop

Assignment:

1. Implement 3 bit ring counter using D flipflop
2. Implement 3 bit Johnson counter using D flipflop

1. Design and implement 2 bit up synchronous counter using JK flipflop

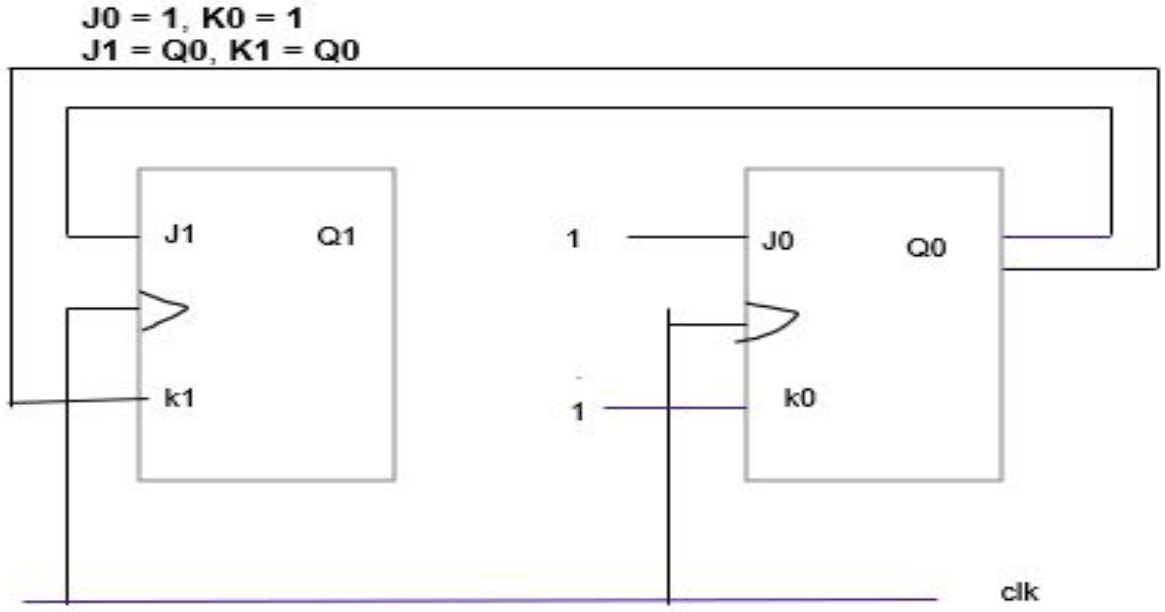
Present (Q1 Q0)	Next ($Q1^+$ $Q0^+$)	inputs			
		J1	K1	J0	k0
00	01	0	X	1	X
01	10	1	x	x	1
10	11	x	0	1	X
11	00	x	1	x	1

Solving using 2 variable K map of present state

J0 = 1, K0 = 1

J1 = Q0, K1 = Q0

Design and implement 2 bit up synchronous counter using JK flipflop



Design and implement 2 bit up synchronous counter using JK flipflop

```
Module jkff (
// JK Flip-Flop with synchronous module
)
// 2-bit synchronous up counter using JK flip-flops
module up_counter_2bit ( input clk,    input reset,    output [1:0] q);
wire j0, k0, j1, k1;
  assign j0 = 1'b1;
assign K0 = 1'b1;
assign j1 = q[0]
   assign k1 = q[0];
// Instantiate two JK flip-flops
  jkff jkff0 (.clk(clk), .reset(reset), .j(j0), .k(k0), .q(q[0]));
  jkff jkff1 (.clk(clk), .reset(reset), .j(j1), .k(k1), .q(q[1]));endmodule
```

Design and implement 2 bit up synchronous counter using JK flipflop

```
module tb_up_counter_2bit;
    reg clk, reset;
    wire [1:0] q;

    // Instantiate counter
    up_counter_2bit uut (.clk(clk),
.reset(reset), .q(q));

    // Clock generation
    initial clk = 0;
    always #5 clk = ~clk;
 // Clock period = 10
```

```
initial begin
        $dumpfile("up_counter_jk.vcd");
        $dumpvars(0, tb_up_counter_2bit);

$display("Time\tClk\tReset\tQ1Q0");
$monitor("%0t\t%b\t%b\t%b%b", $time, clk, reset,
q[1], q[0]);
        reset = 1; #10;   // Apply reset
        reset = 0;
        #60;              // Run counter for a while
        reset = 1; #10;   // Reset again
        reset = 0;
        #40;
        $finish;
    end
endmodule
```

2. Design and implement 2 bit Down synchronous counter using JK flipflop

Q1q0	Q1q0(nextstate)	J1k1	j0k0
11	10	X0	X1
10	01	X1	1x
01	00	0x	X1
00	11	1x	1x

Jo=k0=1

J1k1=q0`

Design and implement 2 bit Down synchronous counter using JK flipflop

Define JK flipflop module

```
// 2-bit synchronous down counter using JK flip-flops
module down_counter_2bit (    input clk,    input reset,    output [1:0] q);
   // JK inputs    wire j0, k0, j1, k1;
assign j0 ,ko values
  assign j[1],k[1] values
  // Instantiate flip-flops
 jkff jkff0(------------------);
  jkff jkff1 (-----------------);
endmodule
```

Design and implement 2 bit Down synchronous counter using JK flipflop-test bench

```
module tb_down_counter_2bit;
 reg clk, reset;    wire [1:0] q;
 // Instantiate counter
  down_counter_2bit uut (.clk(clk),
.reset(reset), .q(q));
// Clock generation
    initial begin
    clk = 0;
   forever #5 clk = ~clk;   // Clock
period = 10
 end
```

```
 initial begin       $dumpfile("down_counter_jk.vcd");
$dumpvars(0, tb_down_counter_2bit);

$display("Time\tClk\tReset\tQ1Q0");
 $monitor("%0t\t%b\t%b\t%b%b", $time, clk, reset,
q[1], q[0]);

     reset = 1; #10;   // Apply reset
   reset = 0;       #80;            // Run counter for a while
reset = 1; #10;   // Reset again
      reset = 0;       #40;
  $finish;
   end
endmodule
```

3. Design and implement 2 bit up -Down synchronous counter using JK flipflop

M(up_down)-up-1, down-0	Q1q0	Q1q0 (Next state)	J1k1	j0k0
1	00	01	0x	1x
1	01	10	1x	X1
1	10	11	X0	1x
1	11	00	X1	X1
0	11	10	X0	X1
0	10	01	X1	1x
0	01	00	0x	X1
0	00	11	1x	1x

J0=k0=1

Ji=k1=Mq0+M`Q0`

Design and implement 2 bit up -Down synchronous counter using JK flipflop

JK flipflop module

module updown_counter_2bit (input wire clk, input wire rst, input wire up_down, // 1: up, 0: down, output wire [1:0] q);

Define:

JO KO

J1 K1

// Flip-flops instantiation

Design and implement 2 bit up -Down synchronous counter using JK flipflop-test bench

```
`timescale 1ns/1ps
module tb_updown_counter_2bit;
 reg clk = 0;
 reg rst = 1;
  reg up_down = 1;
wire [1:0] q;
updown_counter_2bit dut (.clk(clk), .rst(rst),
up_down(up_down), .q(q));
// 10 ns period clock  always #5 clk = ~clk;
```

Design and implement 2 bit up -Down synchronous counter using JK flipflop-test bench

```
/ Print after each synchronous update
  always @(posedge clk) begin
    $display("%0t  clk=%0b rst=%0b upDn=%0b  Q=%0b%0b",
            $time, clk, rst, up_down, q[1], q[0]);
  end

  initial begin
    $dumpfile("updown_2bit_jk.vcd");
    $dumpvars(0, tb_updown_counter_2bit);
```

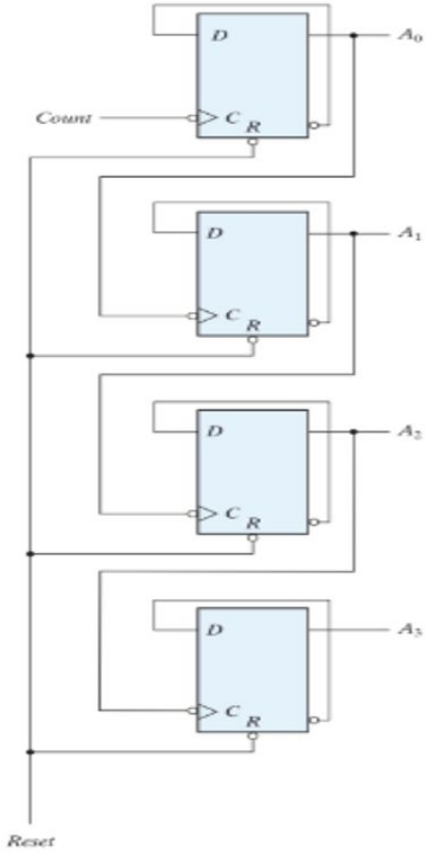
Design and implement 2 bit up -Down synchronous counter using JK flipflop-test bench

```
    // Keep reset high for one edge -> Q becomes 00 at first posedge
     @(negedge clk); rst = 1;
     @(negedge clk); rst = 0;       // release just before a posedge

     // Count UP to reach 11: 00 -> 01 -> 10 -> 11
     up_down = 1;
     repeat (3) @(negedge clk);     // change only on negedge

     // Now count DOWN: 11 -> 10 -> 01 -> 00
     up_down = 0;
     repeat (3) @(negedge clk);
     $finish;
   end
endmodule
```

4. Design and implement ripple 2 bit up counter using D flipflops



4. Design and implement ripple 2 bit up counter using D flipflops

```
// Sequence: 00 → 01 → 10 → 11 → 00 → …
`timescale 1ns/1ps
module dff (
  input  wire clk,
  input  wire rst,   // async reset, active-high
  input  wire d,
  output reg  q
);
  always @(posedge clk or posedge rst) begin
    if (rst) q <= 1'b0;
    else     q <= d;
  end
endmodule
```

4. Design and implement ripple 2 bit up counter using D flipflops

```
module ripple_up_counter_2bit (
  input  wire clk,
  input  wire rst,
  output wire [1:0] q
);
  wire q0, q1;
  // LSB toggles every system clock
  dff ff0 (.clk(clk),  .rst(rst), .d(~q0), .q(q0));

  // MSB toggles when q0 falls (use posedge of ~q0)
  dff ff1 (.clk(~q0),  .rst(rst), .d(~q1), .q(q1));
  assign q = {q1, q0};
endmodule
```

Thank you