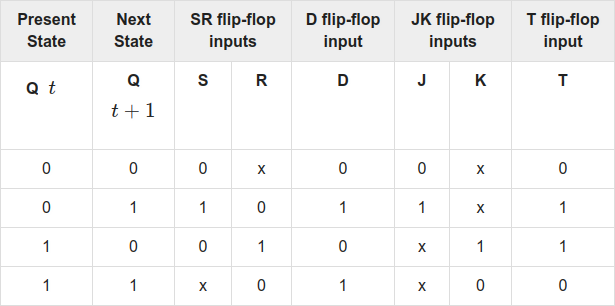
Name : DHVANI PATEL

Roll No:21BCP116 Div 2 -G4

Lab Assignment – 10 : Flip flops Conversion

In the previous lab we discussed about Sequential circuits. We developed Verilog modules for S-R latch and S-R flip-flops. In today's assignment, we will learn about the conversion of one flip-flop into other.

Please use the table 1, describing the input and output states of various flip-flops for your reference.



# Table 1: Input and output states of various flip-flops

**Q1>** The **JK flip flop** is basically a gated SR flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level “1”.

1. Use the table 1 to develop the conversion table for S-R to J-K flip flop and generate the conversion expression between JK and SR via K map.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1. J | K | Qn | Qn+1 | S | R |
| 0 | 0 | 0 | 0 | 0 | X |
| 0 | 0 | 1 | 1 | X | 0 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | X | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **X** | **0** | **0** |
| **1** | **X** | **0** | **1** |

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **0** | **1** | **X** |
| **0** | **0** | **1** | **0** |

S=J’Q R=KQ

1. Use this conversion expression to Modify the S-R flip flop created in last lab into JK flip flop.
2. Verify the functionality of J-K flip flop using suitable Testbench.

Code:

Design:

module JK\_FF\_Behave(J,K, CLK, Q, QBar);

input J,K,CLK;

wire S,R;

output reg Q, QBar;

initial Q = 1;

initial QBar = 0;

and(S,J,QBar);

and(R,K,Q);

always@(posedge CLK)

begin

if(S == 1 & R ==0)

begin

Q = 1;

QBar = 0;

end

else if(S == 0 & R == 1)

begin

Q = 0;

QBar = 1;

end

else if(S == 1 & R == 0)

begin

Q = 1;

QBar = 0;

end

else if(S == 0 & R == 0)

begin

Q <= Q;

QBar <= QBar;

end

end

endmodule

Testbench:

module tb\_JK\_FF\_Behave;

reg J,K,CLK;

wire Q,Q\_bar;

JK\_FF\_Behave al(J,K,CLK,Q,Q\_bar);

initial

begin

J=0; K=0; CLK=0; #5;

$display("J = %b, K = %b, CLK = %b, Q = %b, Q\_bar = %b",J,K,CLK,Q,Q\_bar);

J=0; K=1; CLK=1; #5;

$display("J = %b, K = %b, CLK = %b, Q = %b, Q\_bar = %b",J,K,CLK,Q,Q\_bar);

J=1; K=0; CLK=1; #5;

$display("J = %b, K = %b, CLK = %b, Q = %b, Q\_bar = %b",J,K,CLK,Q,Q\_bar);

J=1; K=1; CLK=1; #5;

$display("J = %b, K = %b, CLK = %b, Q = %b, Q\_bar = %b",J,K,CLK,Q,Q\_bar);

J=0; K=1; CLK=0; #5;

$display("J = %b, K = %b, CLK = %b, Q = %b, Q\_bar = %b",J,K,CLK,Q,Q\_bar);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

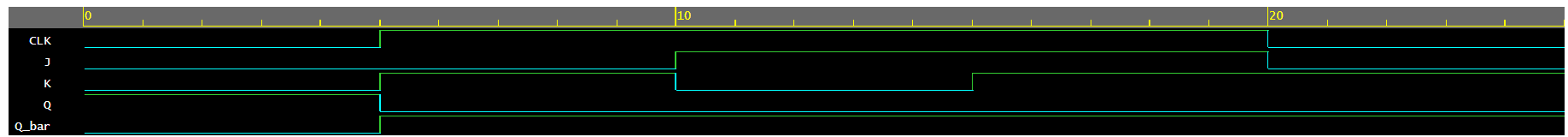
end

endmodule

Output:

VCD info: dumpfile dump.vcd opened for output.  
J = 0, K = 0, CLK = 0, Q = 1, Q\_bar = 0  
J = 0, K = 1, CLK = 1, Q = 0, Q\_bar = 1  
J = 1, K = 0, CLK = 1, Q = 0, Q\_bar = 1  
J = 1, K = 1, CLK = 1, Q = 0, Q\_bar = 1  
J = 0, K = 1, CLK = 0, Q = 0, Q\_bar = 1

Waveform:



Q2> Now lets convert the J-K flip flop created in question 1 into D flip flip. One can use the table 1 to perform similar steps as per previous solution for this conversion.

1. Develop the behavioral verilog module of D flip-flop, converting it from a J-K flip flop. You may utilize if-else statements to develop your verilog code.

|  |  |  |
| --- | --- | --- |
| 1. D flip-flop characteristic table | | |
| D | Q(n) | Q(n+1) |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| JK flip-flop excitation table | | | |
| Q(n) | Q(n+1) | J | K |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

Conversion table [JK -> D] & K-map for J & K

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Conversion table [JK -> D] | | | | |
| D | Q(n) | Q(n+1) | J | K |
| 0 | 0 | 0 | 0 | X |
| 0 | 1 | 0 | X | 1 |
| 1 | 0 | 1 | 1 | X |
| 1 | 1 | 1 | X | 0 |

K-Map for J and K

|  |  |
| --- | --- |
| **0** | **X** |
| **1** | **X** |

|  |  |
| --- | --- |
| **X** | **1** |
| **X** | **0** |

J=D K=D’

1. Verify the functionality via a suitable test bench code.

Code:

Design:

module JK\_FF\_Behave(D,CLK, Q, QBar);

input J,K,CLK ,D;

wire D\_not;

output reg Q, QBar;

initial Q = 0;

initial QBar = 1;

not(D\_not , D);

and(J,D,QBar);

and(K,D\_not,Q);

always@(posedge CLK)

begin

if(J == 0 & K==1)

begin

Q = 0;

QBar = 1;

end

else if(J == 1 & K == 0)

begin

Q = 1;

QBar = 0;

end

end

endmodule

TestBench:

module tb\_JK\_FF\_Behave;

reg CLK ,D;

wire Q,Q\_bar;

JK\_FF\_Behave al(D,CLK,Q,Q\_bar);

initial

begin

D=0; CLK=0; #5;

$display("D = %b, CLK = %b, Q = %b, Q\_bar = %b",D,CLK,Q,Q\_bar);

D=0; CLK=1; #5;

$display("D = %b, CLK = %b, Q = %b, Q\_bar = %b",D,CLK,Q,Q\_bar);

D=1; CLK=0; #5;

$display("D = %b, CLK = %b, Q = %b, Q\_bar = %b",D,CLK,Q,Q\_bar);

D=1; CLK=1; #5;

$display("D = %b, CLK = %b, Q = %b, Q\_bar = %b",D,CLK,Q,Q\_bar);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

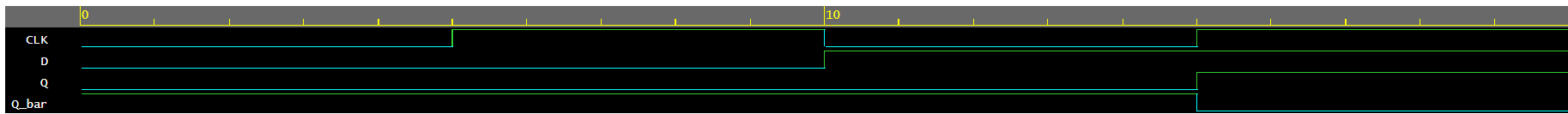
end

endmodule

Output:

VCD info: dumpfile dump.vcd opened for output.  
D = 0, CLK = 0, Q = 0, Q\_bar = 1  
D = 0, CLK = 1, Q = 0, Q\_bar = 1  
D = 1, CLK = 0, Q = 0, Q\_bar = 1  
D = 1, CLK = 1, Q = 1, Q\_bar = 0

Waveform:



Q3> Develop a characteristic table of T flip-flop along with the excitation inputs of JK flip flop from table 1.

1. Use K map to develop the conversion relation between T and JK flip flop.
2. Characteristic table of T flip-flop & excitation table of J-K flip-flop

|  |  |  |
| --- | --- | --- |
| T flip-flop characteristic table | | |
| T | Q(n) | Q(n+1) |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| JK flip-flop excitation table | | | |
| Q(n) | Q(n+1) | J | K |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | X | 0 |

Conversion table [JK -> T] & K-map for J & K

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Conversion table [JK -> T] | | | | |
| T | Q(n) | Q(n+1) | J | K |
| 0 | 0 | 0 | 0 | X |
| 0 | 1 | 1 | X | 0 |
| 1 | 0 | 1 | 1 | X |
| 1 | 1 | 0 | X | 1 |

K-Map for J and K

|  |  |
| --- | --- |
| **0** | **X** |
| **1** | **X** |

|  |  |
| --- | --- |
| **X** | **0** |
| **X** | **1** |

J=T K=T

1. Develop a behavioral verilog module for JK flip flop using case statements. Modify it to act like a T flip flop.
2. Validate the modification via a suitable test bench.

Code:

module jkfft(input t, input clk, output reg q, output reg qnot);

wire j , k;

initial q=0;

assign qnot=1;

and(j,t,qnot);

and(k,t,q);

always @(posedge clk)

case ({j, k})

2'b00: q<=q;

2'b01: q<=1'b0;

2'b10: q<=1'b1;

2'b11: q<=~q;

endcase

endmodule

TestBench:

module tb\_jkfft;

reg CLK ,T;

wire Q,Q\_bar;

jkfft al(T,CLK,Q,Q\_bar);

initial

begin

T=0; CLK=0; #5;

$display("T = %b, CLK = %b, Q = %b, Q\_bar = %b",T,CLK,Q,Q\_bar);

T=0; CLK=1; #5;

$display("T = %b, CLK = %b, Q = %b, Q\_bar = %b",T,CLK,Q,Q\_bar);

T=1; CLK=0; #5;

$display("T = %b, CLK = %b, Q = %b, Q\_bar = %b",T,CLK,Q,Q\_bar);

T=1; CLK=1; #5;

$display("T = %b, CLK = %b, Q = %b, Q\_bar = %b",T,CLK,Q,Q\_bar);

end

initial

begin

$dumpfile("dump.vcd");

$dumpvars(1);

end

endmodule

Output:

T = 0, CLK = 0, Q = 0, Q\_bar = 1  
T = 0, CLK = 1, Q = 0, Q\_bar = 1  
T = 1, CLK = 0, Q = 0, Q\_bar = 1  
T = 1, CLK = 1, Q = 1, Q\_bar = 1  
Finding VCD file...

Waveform:

