HACETTEPE UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING BBM233

Name : Mehmet Taha, Mehmet Emin, Alperen

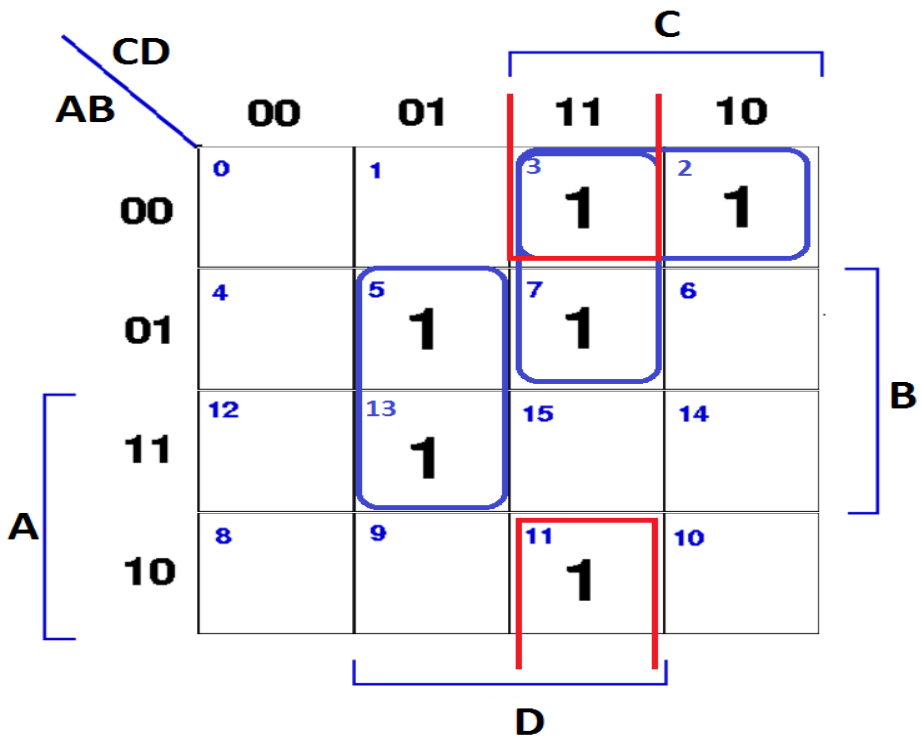
Surname : Usta, Akoğuz, Ekinci

Number : 21527472,21626895,21627188

Subject : Design 4 bit prime number counter with JK flip flop(s)

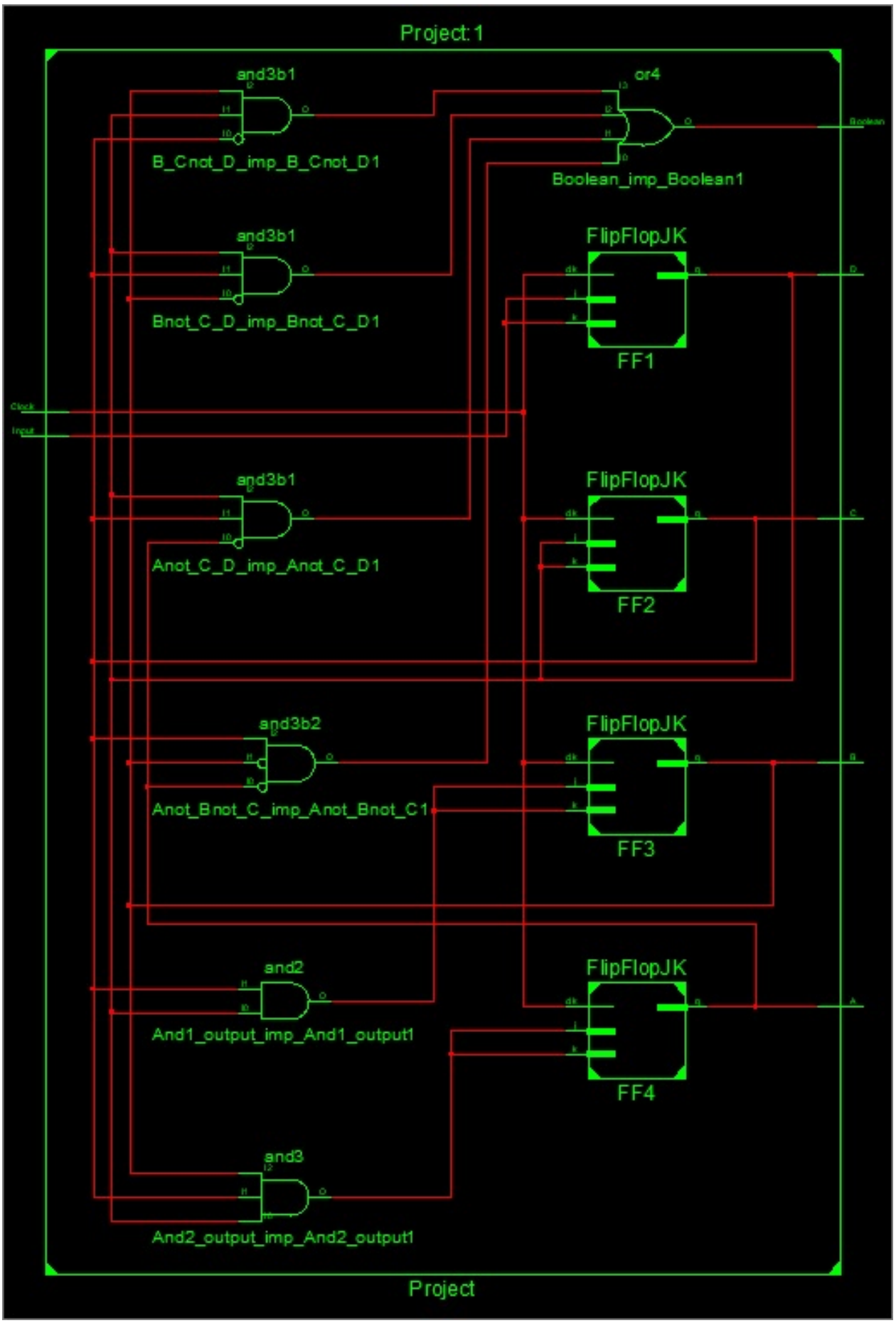
Truth Table

	A	B	C	D	BOOLEAN
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0



Groups

(2,3)	$\bar{A}.\bar{B}.C$
(3,7)	$\bar{A}.C.D$
(3,11)	$\bar{B}.C.D$
(5,13)	$B.\bar{C}.D$


Project:1
and3b1
B_Cnot_D_imp_B_Cnot_D1
or4
Boolean_imp_Boolean1
Boolean
and3b1
Bnot_C_D_imp_Bnot_C_D1
FlipFlopJK
FF1
D
Clock
Input
and3b1
Anot_C_D_imp_Anot_C_D1
FlipFlopJK
FF2
C
and3b2
Anot_Bnot_C_imp_Anot_Bnot_C1
FlipFlopJK
FF3
B
and2
And1_output_imp_And1_output1
FlipFlopJK
FF4
A
and3
And2_output_imp_And2_output1
Project

We defined a standart JK-FlipFlop. Then used it to make a 4-bit Up Counter.From the K-map of the prime numbers we found the functions fort he numbers and implemented them as AND/OR Gates.

We printed Each cycle, thus got prime numbers with the "Boolean" value equal to 1.

```
module FlipFlopJK(j,k,clk,q);

input j,k;
input clk;
output q;
reg q;

initial
begin q=1'b0;
q_not=1'b1;
end
always @ (posedge clk)
begin
case({j,k})
{1'b0,1'b0}:begin q=q; end
{1'b0,1'b01}:begin q=1'b0; end
{1'b1,1'b0}:begin q=1'b1; end
{1'b1,1'b1}:begin q=~q; end
endcase
end
endmodule

module Project(Input, Clock ,Boolean,A,B,C,D);
  input Input, Clock;
  output Boolean;
  output A,B,C,D;
  wire   A,B,C,D,And1_output, And2_output,Anot_D, Anot_Bnot_C, Cnot_D_B, A_Bnot_C_D;

 //4-Bit Counter:
 FlipFlopJK FF1 (Input, Input, Clock, A);
 FlipFlopJK FF2 (A, A, Clock, B);
 and And1 (And1_output, A, B);
 FlipFlopJK FF3 (And1_output, And1_output, Clock, C);
 and And2 (And2_output, A, B, C);
 FlipFlopJK FF4 (And2_output, And2_output, Clock, D);

 //Calculations:
 and And3 (Anot_D, ~A, D);
 and And4 (Anot_Bnot_C, A, B, C);
 and And5 (Cnot_D_B, ~C, D, B);
 and And6 (A_Bnot_C_D, A, ~B, C, D);
 or Or1 (Boolean, Anot_D, Anot_Bnot_C, Cnot_D_B, A_Bnot_C_D);

endmodule
```