LAB REPORT-B20EE016

Digital Design - EEL2020

Dhruv

B20EE016

Asynchronous Reset D flip flop

TRUTH TABLE

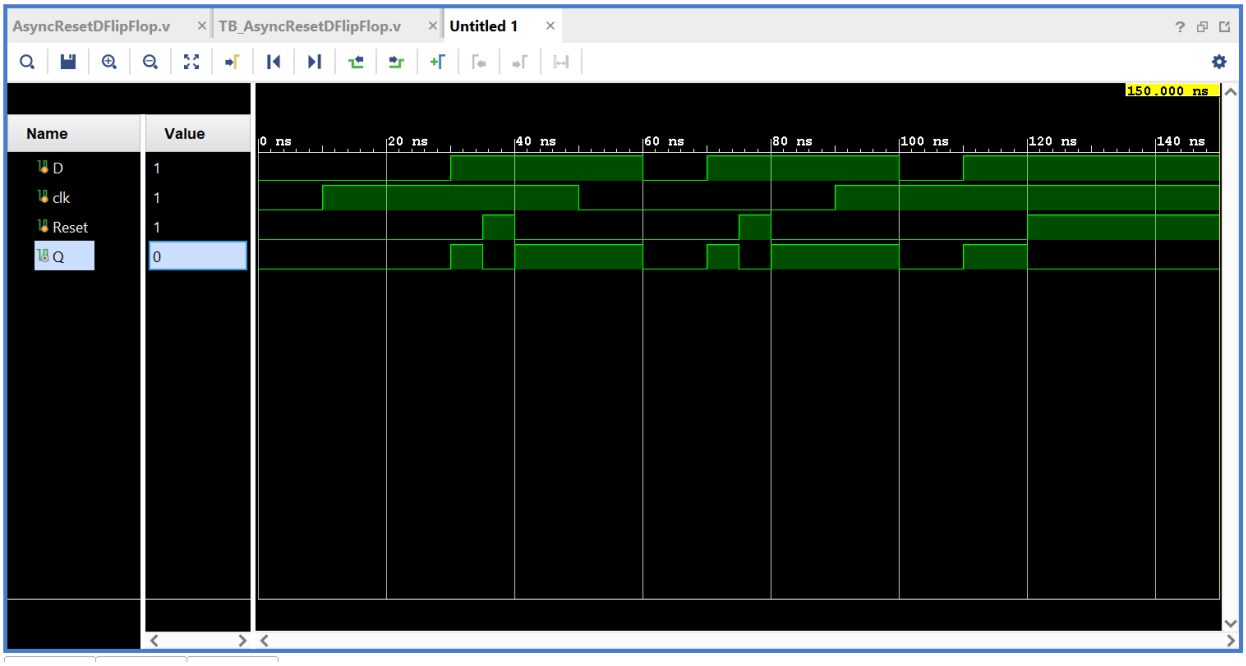
INPUTS				OUTPUTS	
$\overline{PR}$	$\overline{CLR}$	CLK	D	Q	$\overline{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	X	X
1	1	↑	1	1	0
1	1	↑	0	0	1
1	1	0	X	Q_0	$\overline{Q}_0$

Code

```
`timescale 1ns / 1ps

module AsyncResetDFlipFlop(
input D,Reset,clk,
output reg Q
);
always @(posedge clk,Reset)
    begin
        if(Reset) Q=1'b0;
        else
        Q=D;
    end
endmodule
```

Simulation



Test Bench

```
`timescale 1ns / 1ps
module TB_AsyncResetDFlipFlop();

reg D,clk,Reset;
wire Q;

AsyncResetDFlipFlop A1(D,Reset,clk,Q);
initial
begin
    //default
    D=1'b0;clk=1'b0;Reset=1'b0;
    #10 clk=1'b1;            //clock 1 postive edge
    #10 D=1'b0;  //d=0 and reset
    #10 D=1'b1;
    #5 Reset=1'b1;  //d=0 and reset
    #5 Reset=1'b0;

    #10 clk=1'b0;
    #10 D=1'b0;
    #10 D=1'b1;
    #5 Reset=1'b1;  //d=0 and reset
    #5 Reset=1'b0;
```

```
    #10 clk=1'b1;
    #10 D=1'b0;
    #10 D=1'b1;
    #10 Reset=1'b1;

end
initial #150 $finish;
endmodule
```

Synchronous Reset D flip flop

Code

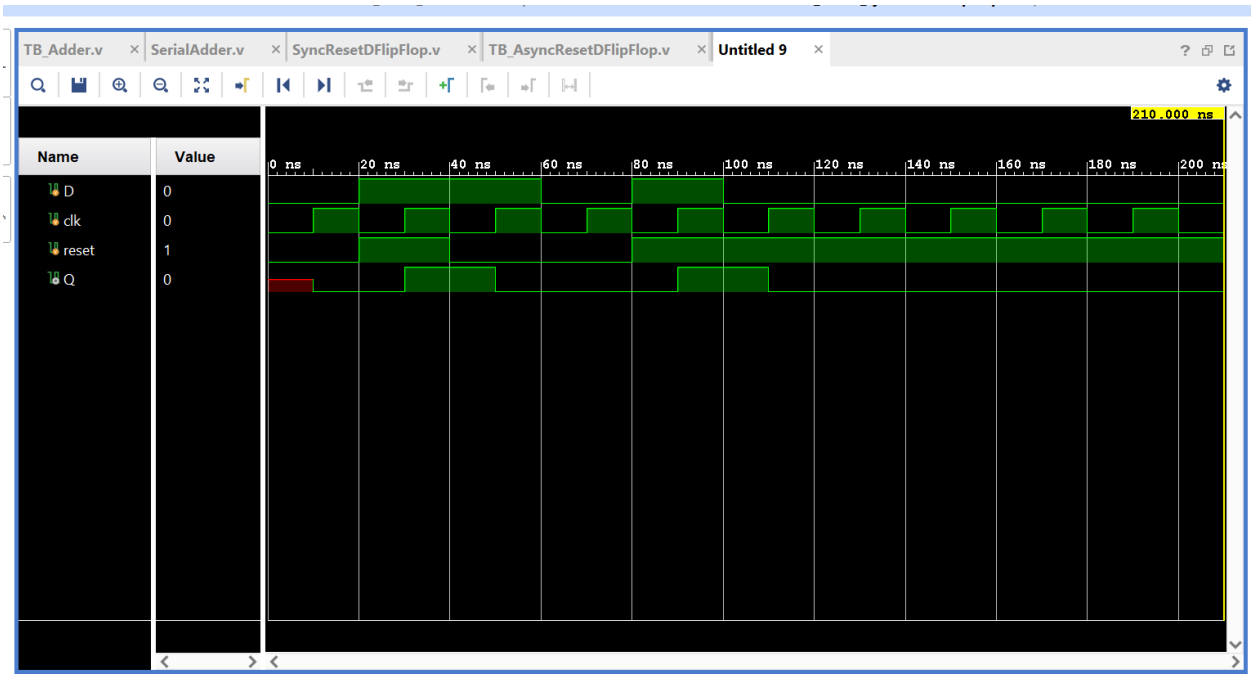
```
`timescale 1ns / 1ps

module SyncResetDFlipFlop(
input D,clk,reset,
output reg Q);

always@(posedge clk)
    begin
        if(!reset)
        begin
             Q <=1'd0;
        end
        else
        begin
             Q<=D;
        end
    end

endmodule
```

Simulation



Test Bench

```
`timescale 1ns / 1ps
module TB_syncResetDFlipFlop();
reg D;
reg clk,reset;
wire Q;
SyncResetDFlipFlop A2(D,clk,reset,Q);
initial begin
    D = 0;clk = 0;reset=0;
    end
    always #10 clk=~clk;
    initial
    begin
        #20 D= 1'b1;reset = 1'b1;
        #20 D = 1'b1;reset=1'b0;

        #20 D = 1'b0;
        #20 D= 1'b1;
        reset=1'b1;
        #20 D = 1'b0;
        #110 $stop;
    end
endmodule
```

Blocking Operation Verilog Code:

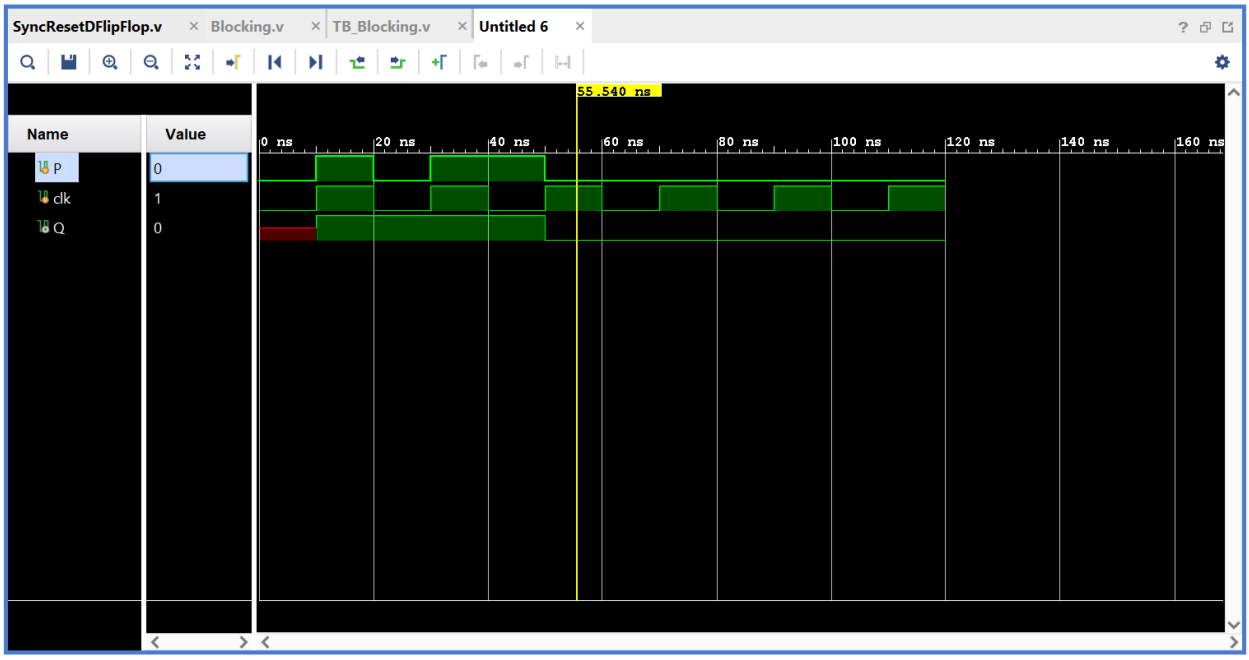
Code

```
`timescale 1ns / 1ps

module Blocking(
input p,clk,
output reg q);

reg D1;
always@(posedge clk)
begin
D1 = p;
q = D1;
end
endmodule
```

Simulation



Test Bench

```
`timescale 1ns / 1ps

module TB_Blocking();
reg P,clk;
wire Q;
Blocking B(P,clk,Q);
initial begin
P = 0;
clk = 0;
end
always #10 clk=~clk;
initial
begin
    #10 P = 1'b1;

    #10 P = 1'b0;
    #10 P = 1'b1;
    #10 P = 1'b1;
    #10 P = 1'b0;
    #70 $stop;
end
endmodule
```

Non-Blocking Operation verilog Code:

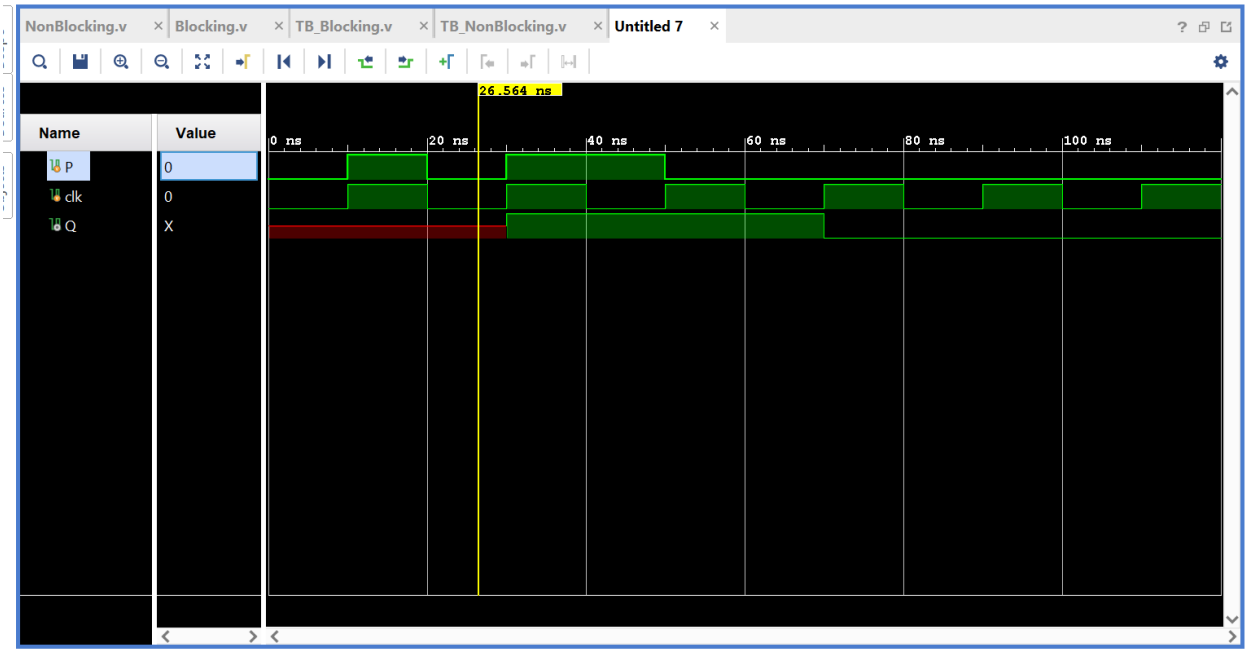
Code

```
`timescale 1ns / 1ps

module NonBlocking(
input p,clk,
output reg q
    );

reg D1;
always@(posedge clk)
begin
D1 <= p;
q <= D1;
end
endmodule
```

Simulation



Test-Bench

```
`timescale 1ns / 1ps
module TB_NonBlocking();
reg P,clk;
wire Q;
NonBlocking B(P,clk,Q);
initial begin
P = 0;
clk = 0;
end
always #10 clk=~clk;
initial
begin
    #10 P = 1'b1;

    #10 P = 1'b0;
    #10 P = 1'b1;
    #10 P = 1'b1;
    #10 P = 1'b0;
    #70 $stop;
end
endmodule
```

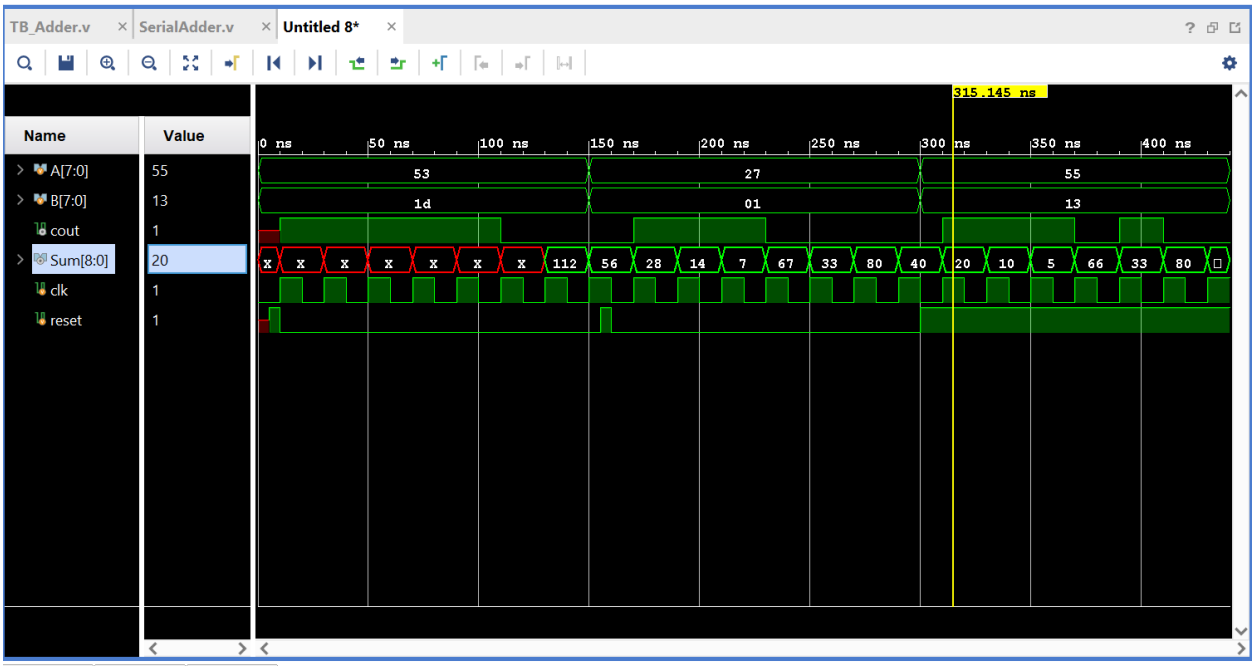
8 bit serial addition

Code

```
`timescale 1ns / 1ps
module SerialAdder(
input [7:0]A,B,
input clk,reset,
output reg cout,
output reg [8:0]sum
);

    reg [7:0]Atemp;
    reg [7:0]Btemp;
    reg cin;
    always@(posedge reset)begin
    if(reset == 1) begin
    cin = 0;
    Atemp[7:0] = A[7:0];
    Btemp[7:0] = B[7:0];
    end
    end
        always@(posedge clk)
        begin
        sum[7] = Atemp[0] ^ Btemp[0] ^ cin;
        sum = sum >> 1;
        cout = (Atemp[0] & Btemp[0]) | (cin & Atemp[0]) | (  cin  &Btemp[0] );
        Atemp = Atemp>>1;
        Btemp = Btemp>>1;
        cin = cout;
    end
endmodule
```

Simulation



Test-Bench

```
`timescale 1ns / 1ps
module TB_Adder();
    reg [7:0]A,B;
    wire cout;
    wire [8:0]Sum;
    reg clk,reset;
    SerialAdder SA(A,B,clk,reset,cout,Sum);
    initial begin
    clk = 0;
    end
    always #10 clk = !clk;
    initial begin
    A=8'b01010011;B=8'b00011101;
    #5 reset=1'b1;#5 reset=1'b0;

    #140 A=8'b00100111;B=8'b00000001;
    #5 reset=1'b1;#5 reset=1'b0;

    #140 A=8'b01010101;B=8'b00010011;reset=1'b1;
    #140 $stop;
    end
```