CASEST

हैदराबाद विश्वविद्यालय
University of Hyderabad

Lab Assignment

111 Sequence detector, Full adder, Johnson counter

Submitted by:

Full Name **Enrolment No.**

Abhinav M 23PMMT11(@uohyd.ac.in)

Some screenshots in this report are missing due to computer malfunction as a result of which old code and screenshots vanished.

17-04-2024

Centre for Advanced Studies in Electronics Science and Technology

111 Sequence detector

Simulate and verify 111 sequence detector using ILA

Code

```
module seq_dect (clk, rst, in, out);
input [2:0] in;
output reg out;
input clk, rst;
reg [1:0] cs, ns;
parameter s0=2'b00, s1=2'b01, s2=2'b10; always@(posedge clk)
begin
if (!rst) cs<-s0;
else cs<=ns;
end
always@(cs, in) begin case (cs)
s0: begin

if (in==0) begin
ns=s0;
out=0;
end
else
begin
ns=s1;
out=0;
end
end
s1: begin
if (in==0)
begin
end
ns=s0; out=0;
else begin ns-s2;
out=0;
end
end
s2:begin
if(in==0) begin
ns-s0;
out=0;
end
else begin
ns=s0;
out=1;
end
end
endcase
end
endmodule
```

Testbench

```
module seq_dect_tb(); reg clk, rst;
reg [2:0] in;
wire out;
seq_dect uut (clk, rst, in, out);
initial
begin
clk=1'b1;
forever #1 clk=~clk;
end
initial
begin
in=0;
rst=0;
#2 rst=1;
@(posedge clk) in=0;
@(posedge clk) in=1;
@(posedge clk) in=1;
@(posedge clk) in=0;
@(posedge clk) in=1;
@(posedge clk) in=1;
@(posedge clk) in=1;
@(posedge clk) in=1;
@(posedge clk) in=1;
#30 $finish;
end
endmodule
```

Result

Simulated and implementation verified successfully in ZedBoard

Full Adder

Simulate and verify full adder using VIO

Wrapper

```
module fa_vio (clk); input clk;
wire s_1, cout_1, a_1,b_1, cin_1;
);
vio_your_instance_name (
.clk (clk),
.probe_ino (s_1), .probe_inl (cout_1),
// input wire clk
// input wire [0: 0] probe_ine // input wire [0: 0] probe_inl
probe_outo (a_1), // output wire [0: 0] probe_out probe_outl (b_1), //
output wire [0: 0] probe_outl probe_out2(cin_1) // output wire [0: 0]
probe_out2
adder_fa uut(.s (s_1), .cout (cout_1),.a (a_1), .b (b_1), .cin (cin_1));
endmodule
```

Code

```
module adder_fa(s, cout, a, b, cin);
input a, b, cin;
output s, cout;
assign s = a ^ b ^ cin;
assign cout(a& b) | (b & cin) | (a & cin);
endmodule4
```

Testbench

```
module adder_fa_tb();
reg a,b,cin;
wire s,cout;
adder_fa uut (s, cout, a, b, cin);
initial
2 begin
#10 a=0;b=0; cin=0;
#10 a=0;b=0; cin=1;
#10 a=0;b=1; cin=0; #10 a=0;b=1;cin=1; #10 a=1;b=0; cin=0; #10 a=1;b=0;
cin=1; #10 a=1;b=1; cin=0;
#10 a=1;b=1; cin=1;
#10 $finish; end
endmodule
```

Result

Simulated and implementation verified successfully in ZedBoard

Johnson Counter

Simulate and verify Johnson Counter using ILA

Code

```
module jhoson_counter (clk, rst, out);
input clk, rst;
output [3:0]
reg [3:0] q;
out;
always@(posedge clk)
begin
if (rst)
q<=4'b1000;
else
begin
q[3]<=~q[0];
q[2]<=q[3];
q[1]<=q[2];
q[0]<=q[1];
end
end
assign out=q;
endmodule
```

Testbench

```
module
jhonson_counter_tb();
reg clk, rst;
wire [3:0] out;
jhoson_counter uut (clk, rst, out);
initial
begin
clk=0;
forever #5 clk-~clk;
end
initial
begin
rst=1'b1;
#10 rst=1'b0;
#200 $finish;
end
endmodule
```

Result

Simulated and implementation verified successfully in ZedBoard