**San Jose State University**

**Department of Computer Engineering**

**CMPE 200 Report**

**Assignment 1 Report**

**Title** System-Level Design Review

**Semester:** Fall 2023 **Date:** 09/16/2023

**by**

**Name:** Sai Kashyap Kurella **SID:** 016018925

**FACTORIAL SYSTEM TOP LEVEL**

A person standing next to a rectangular board

Description automatically generated

**Fig.1 Top level view of Factorial System**

**SYSTEM DATA PATH**

**A diagram of a computer generated diagram

Description automatically generated with medium confidence**

**Fig.2 System Data Path with Control unit**

**Finite State Machine for Control Unit**

**A diagram of a diagram

Description automatically generated**

**Fig.3 State Transitions of Control unit**

**ASM chart for Control Unit**

**A diagram of a flowchart

Description automatically generated**

**Fig.4 State transitions in ASM chart**

**Verilog Codes of Individual Modules**

**Counter**

module CNT # (parameter N=4)

(

input clk,

input reset,

input en,

input load\_cnt,

input [N-1:0] load\_value,

output[N-1:0] count\_q

);

reg[N-1:0] load\_ff;

//Store the load Value whenever load\_cnt is seen

always@(posedge clk or posedge reset)

begin

if(reset)

begin

load\_ff <= 4'h0;

end

else if(load\_cnt)

begin

load\_ff <= load\_value;

end

else

begin

load\_ff <= load\_ff;

end

end

reg[3:0] count\_ff;

reg[3:0] nxt\_count;

always@(posedge clk or posedge reset)

begin

if(reset)

begin

count\_ff <= 4'hF;

end

else

begin

if(en)

begin

count\_ff <= nxt\_count;

end

else

begin

count\_ff <= count\_ff;

end

end

end

assign nxt\_count = load\_cnt ? load\_value : (count\_ff == 4'h0) ? load\_ff : count\_ff - 4'h1;

//assign nxt\_count = (count\_ff == 4'h0) ? load\_ff : count\_ff - 1'h1;

assign count\_q = count\_ff;

endmodule

**Comparator**

module CMP #(parameter N = 4)

(

input [N-1:0] A,

input [N-1:0] B,

output G

);

assign G = (A>B) ? 1'b1: 1'b0;

endmodule

**Multiplier**

module MUL(input[3:0] A, input[31:0] B, output[31:0] O);

assign O = A\*B;

endmodule

**Multiplexer**

module MUX #(parameter N =1 )

(

input [31:0] A,

input [31:0] B,

input S,

output [31:0] Y

);

assign Y = (S) ? B : A ;

endmodule

**Data register with a load control signal**

module REG #(parameter N = 4)

(

input clk,

input reset,

input load\_reg,

input [31:0] reg\_d,

output [31:0] reg\_q

);

reg[31:0] store\_reg;

//Sequential Block

always@(posedge clk or posedge reset)

begin

if(reset)

begin

store\_reg <= 32'h1;

end

else

begin

if(load\_reg)

begin

store\_reg <= reg\_d;

end

else

begin

store\_reg <= 32'h1;

end

end

end

//Combinational Block

assign reg\_q = store\_reg;

endmodule

**Control Unit**

module Control

(

input clk,

input reset,

input Go,

input G1,

input G12,

output reg Error,

output reg load\_cnt,

output reg Done\_cntrl,

output reg load\_reg,

output reg count

);

parameter S0 = 2'b00, S1 = 2'b01,S2 =2'b10,S3=2'b11;

reg[1:0] present\_state;

reg[1:0] next\_state;

//Sequential Part

always@(posedge clk or posedge reset)

begin

if(reset)

begin

present\_state <= S0;

end

else

begin

present\_state <= next\_state;

end

end

//Combinational Part

always@(\*)

begin

case(present\_state)

S0 : begin // Initial Stage

Error = 1'b0;

load\_cnt = 1'b0;

Done\_cntrl = 1'b0;

count = 1'b0;

load\_reg =1'b0;

if(Go)

begin

next\_state = S1;

end

else

begin

next\_state = S0;

end

end

S1 : begin // Check if input is > 12

if(G12)

begin

next\_state = S0;

Error = 1'b1;

end

else

begin

next\_state = S2;

load\_cnt = 1'b1;

count = 1'b1;

end

end

S2 : begin //Calculation Stage

load\_cnt = 1'b0;

if(G1)

begin

next\_state = S2;

load\_reg = 1'b1;

end

else

begin

next\_state = S3;

count = 1'b0;

end

end

S3 : begin //Done

next\_state = S0;

Done\_cntrl = 1'b1;

end

endcase

end

endmodule

**Top Level module**

`include "CMP.v"

`include "CNT.v"

`include "MUL.v"

`include "REG.v"

`include "MUX.v"

`include "Control.v"

module Top #(parameter N = 4) (

input clk,

input reset,

input Go,

input [N-1:0] fact\_in,

output [31:0] fact\_out,

output Done,

output Error

);

wire load\_cnt; //Control

wire load\_reg; //Control

wire count; //Control

wire[N-1:0] count\_q;

wire G1; //Control

wire[31:0] reg\_d;

wire[31:0] reg\_q;

wire[31:0] O;

wire G12; //Control

wire done; //Control

//Comparator-1 (for checking >1)

CMP c1(.A(count\_q),.B(4'h1),.G(G1));

//Counter

CNT cnt1(.clk(clk),.reset(reset),.en(count),.load\_cnt(load\_cnt),.load\_value(fact\_in),.count\_q(count\_q));

//Multiplier

MUL mul1(.A(count\_q),.B(reg\_q),.O(O));

//Data register with a load control signal

REG reg1(.clk(clk),.reset(reset),.load\_reg(load\_reg),.reg\_d(reg\_d),.reg\_q(reg\_q));

//Multiplexer-1

MUX mux1(.A(O),.B(32'h1),.S(load\_cnt),.Y(reg\_d));

//Multiplexer-2 (Buffer)

MUX mux2(.A(32'h0),.B(reg\_q),.S(done),.Y(fact\_out));

//Comparator-2 (for checking >12)

CMP c2(.A(fact\_in),.B(4'hC),.G(G12));

//Controller

Control ctrl(.clk(clk),.reset(reset),.Go(Go),.G1(G1),.G12(G12),.Error(Error),.load\_cnt(load\_cnt),.load\_reg(load\_reg),.Done\_cntrl(done),.count(count));

assign Done = done;

endmodule

**Top Level Test Bench**

module Top\_tb#(parameter N = 4)();

reg clk;

reg reset;

reg Go;

reg [N-1:0] fact\_in;

wire[31:0] fact\_out;

wire Done;

wire Error;

Top t1(.clk(clk),.reset(reset),.Go(Go),.fact\_in(fact\_in),.fact\_out(fact\_out),.Done(Done),.Error(Error) );

// Clock Generation

always

begin

clk = 1'b1;

#2;

clk = 1'b0;

#2;

end

// Stimulus

initial

begin

reset = 1'b1;

fact\_in = 4'hC;

Go = 1'b0;

@(posedge clk);

reset = 1'b0;

Go = 1'b1;

@(posedge clk);

Go = 1'b0;

#100;

fact\_in = 4'h8;

@(posedge clk);

Go =1'b1;

@(posedge clk);

Go = 1'b0;

#100;

fact\_in = 4'hF;

@(posedge clk);

Go =1'b1;

@(posedge clk);

Go = 1'b0;

#100;

$finish;

end

initial

begin

$monitor ("Result = %b", fact\_out);

$dumpfile ("factorial\_waves.vcd");

$dumpvars();

end

endmodule

**Simulations:**

1. **For fact\_in = 12**

A screenshot of a computer

Description automatically generated

1. **For fact\_in = 8**

A screenshot of a computer

Description automatically generated

1. **For fact\_in = 16**

A screenshot of a computer

Description automatically generated

**Conclusion**

The data-path and the control unit for the factorial system has been designed and verified for all the cases as shown above. All the verilog design files have been submitted along with the report.