

Factorial Calculator:

# Introduction:

The purpose of this lab was to create a factorial calculator using Xilinx Vivado Design Suite through Verilog programming. This module takes a 4-bit binary value and outputs the factorial on a 7-segment display. The module is only able to process factorials as high as 12!, and will give an error message if any the input is a larger number than 12.

The list of tasks were fully completed in lab:

- Waveform validation of factorial

- Design system datapath using basic building blocks

- Design block diagrams

- Construct output table

- Hardware validation using Nexys4 DDR board

- Stimuli and Hardware Verification of factorial

The list of tasks were not fully completed in lab but were finished afterword:

- Simulation and hardware verification of the error flag

# Design Methodology:

The lab assignment called for a hierarchical design consisting of several modules:

## factorial\_top:

This module takes a 4-bit input, a go signal, and a reset signal, and outputs a done signal, a 32-bit product, and an error flag. The input is processed by the datapath while the go signal is processed by the control unit. The product is generated by the datapath while the done and error flags are generated by the control unit.

## control\_unit:

This module takes in go and reset signals from the top module, as well a greater than signal, and a greater than 12 signal from the datapath to generate next state logic. The control unit has 5 states. State 0 is an idle state waiting for the command to begin. State 1 prepares the datapath to begin calculating the factorial. State 2 is a wait state for the internal multiplier to finish its calculations. State 3 is the done state where an output enable is sent to the datapath and the done and error flags will be read. State 4 is a transition state where counter and register values of the datapath are updated.

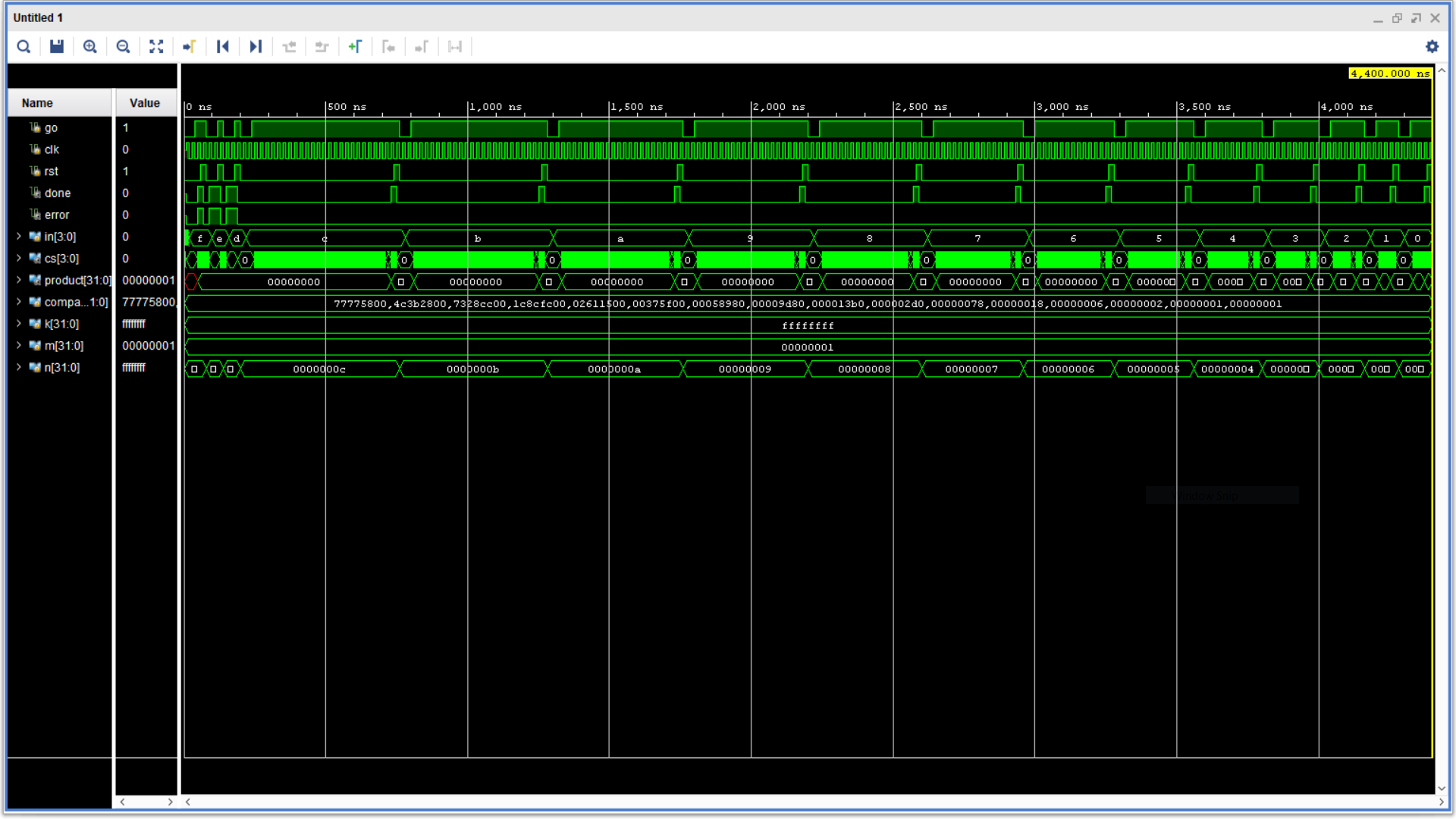
## datapath:

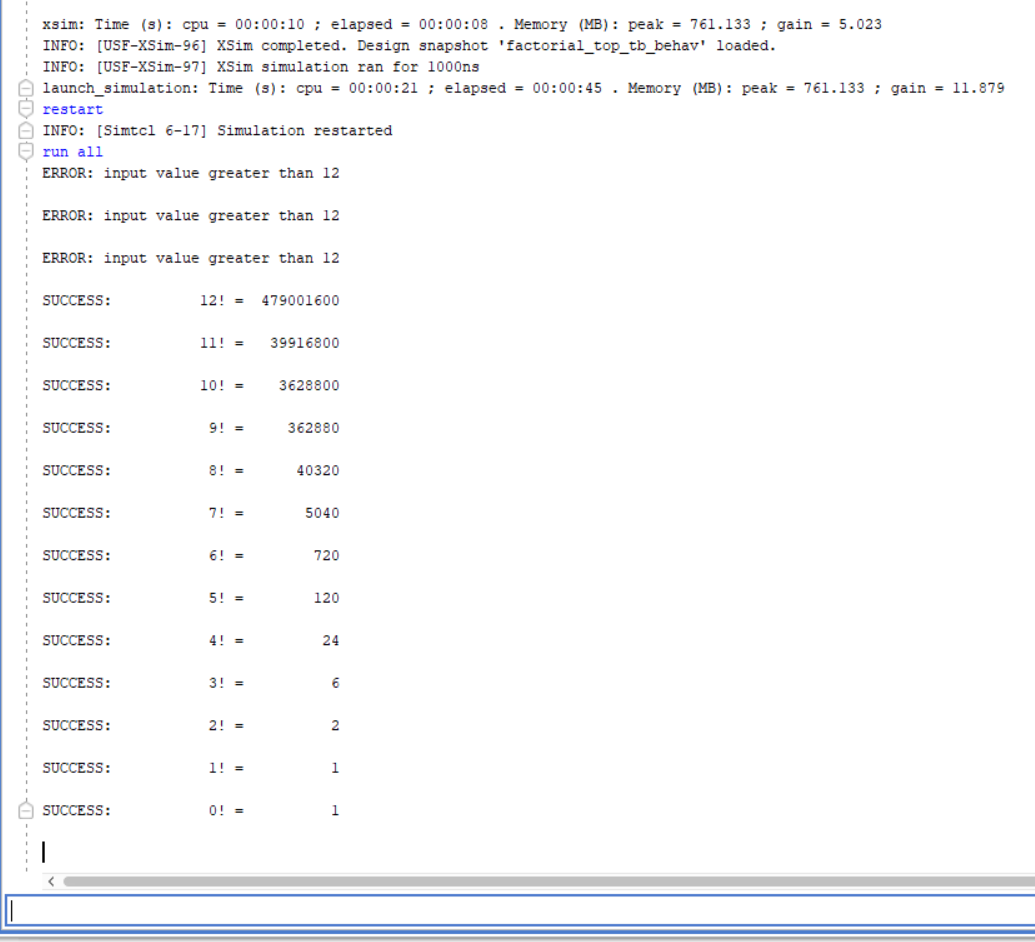
This module takes a 4 bit input from the top module, as well as load counter, enable counter, load register, and output enable signals from the control unit, and outputs a 32-bit product to the top module as well as a greater than signal and a greater than 12 signal to the control unit.

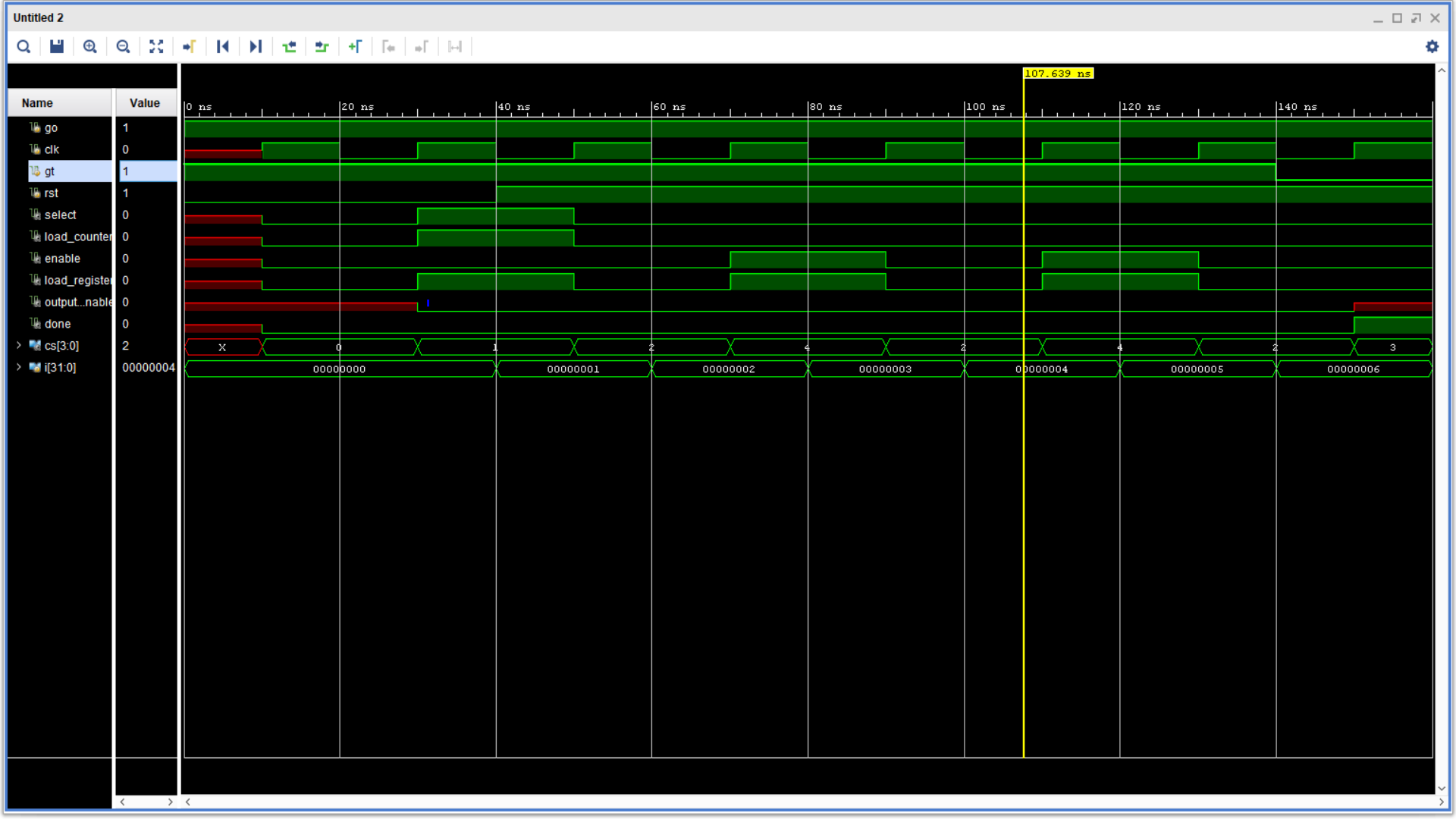
# Procedure:

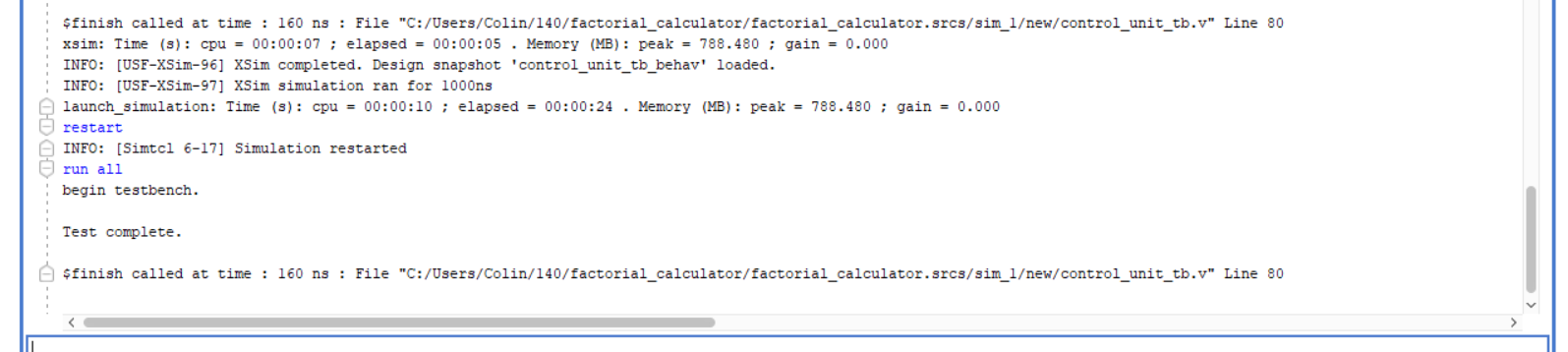
1. Create *factorial* project in Vivado and select the proper hardware: xc7a100csg324-1.
2. Create *factorial\_top.v, control\_unit.v, factorial\_dp.v*, and self-checking testbenches *factorial\_top\_tb.v, control\_unit\_tb.v,* and *factorial\_dp\_tb.v*.
3. Run simulations and verify functionality performs as desired.
4. Create and design *factorial\_FPGA.v, factorial\_FPGA.xdc,*  and all modules necessary for board functionality.
5. Run Synthesis, Implementation, and Bitstream Generation, and program the Nexys 4 FPGA board.
6. Verify board Functionality is performing as desired.

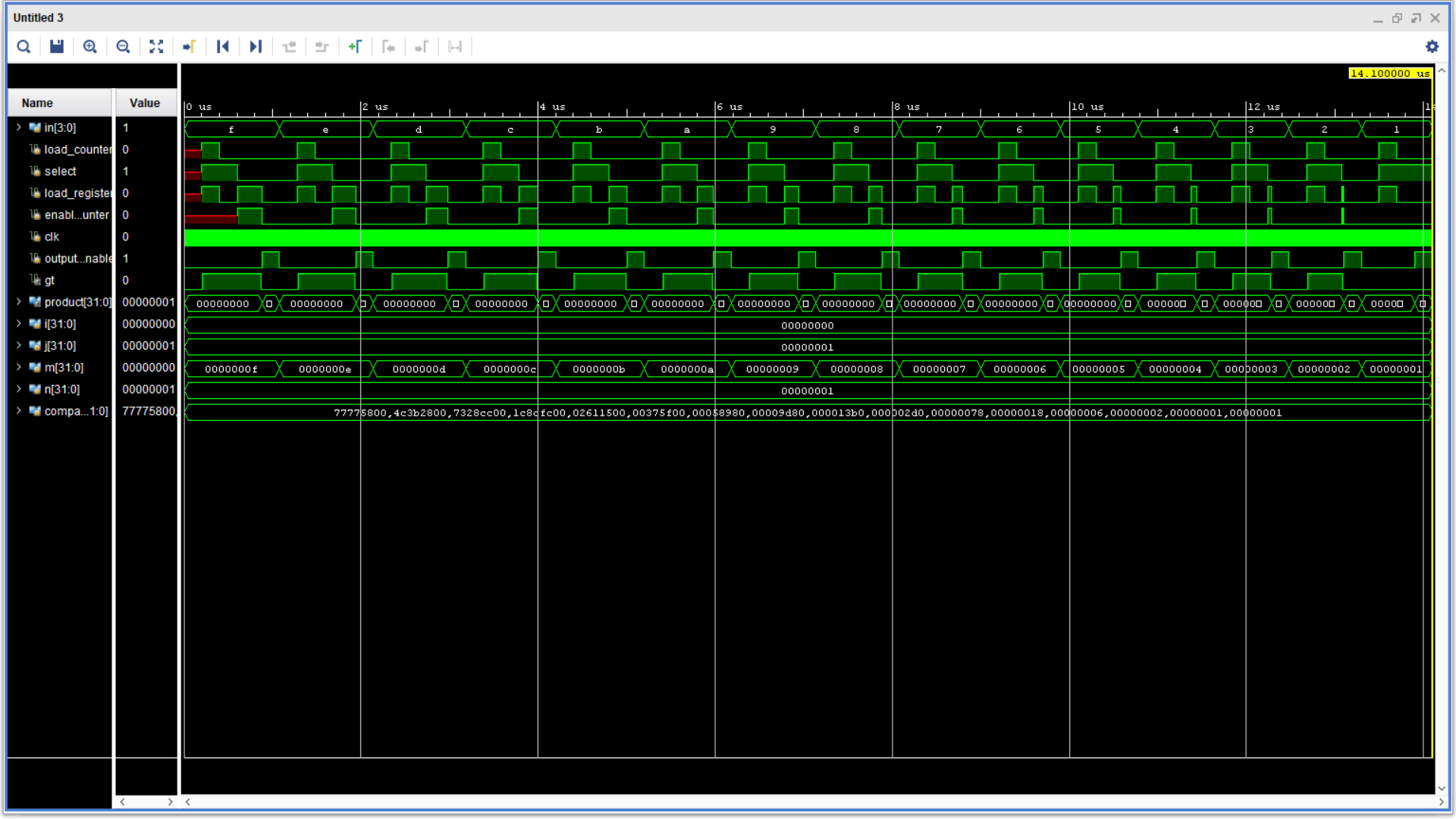
# Simulation Results:

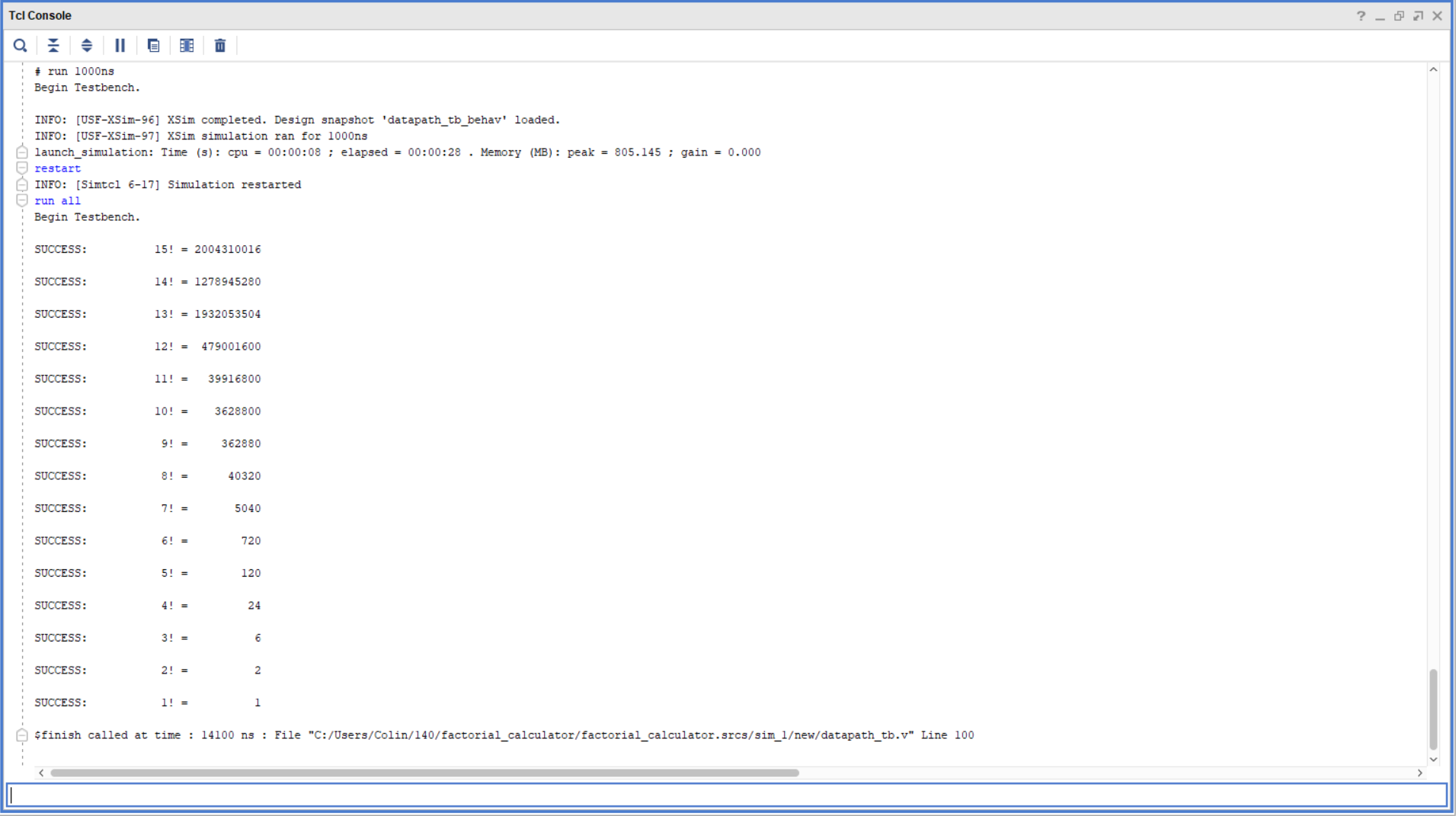
**Figure 1a:** factorial\_top\_tb.v simulation result.

  
**Figure 1b:** factorial\_top\_tb.v Tcl console output.

  
**Figure 2a:** control\_unit\_tb.v simulation results.

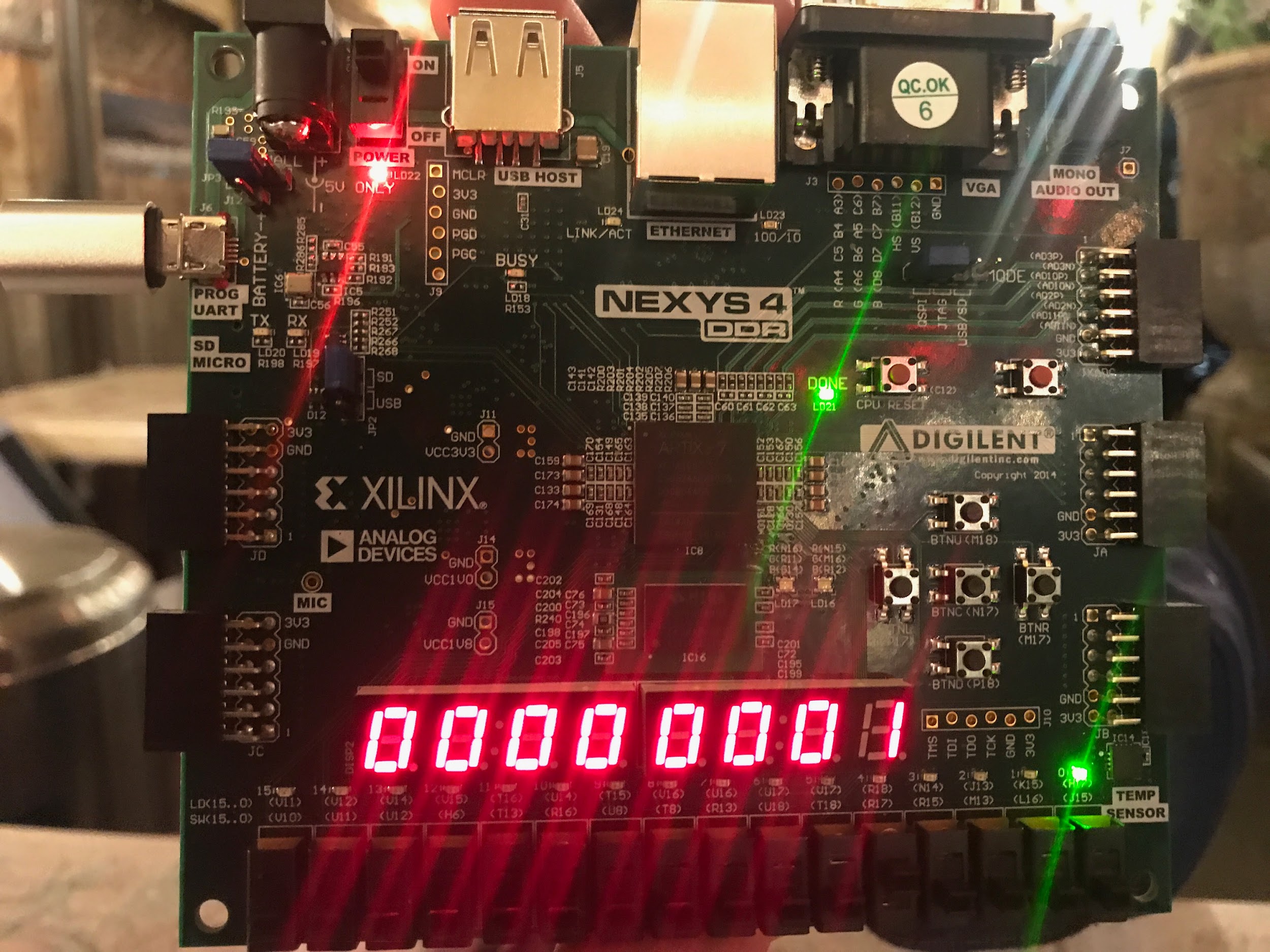
  
**Figure 2b:** control\_unit\_tb.v Tcl console output.

  
**Figure 3a:** datapath\_tb.v simulation results.

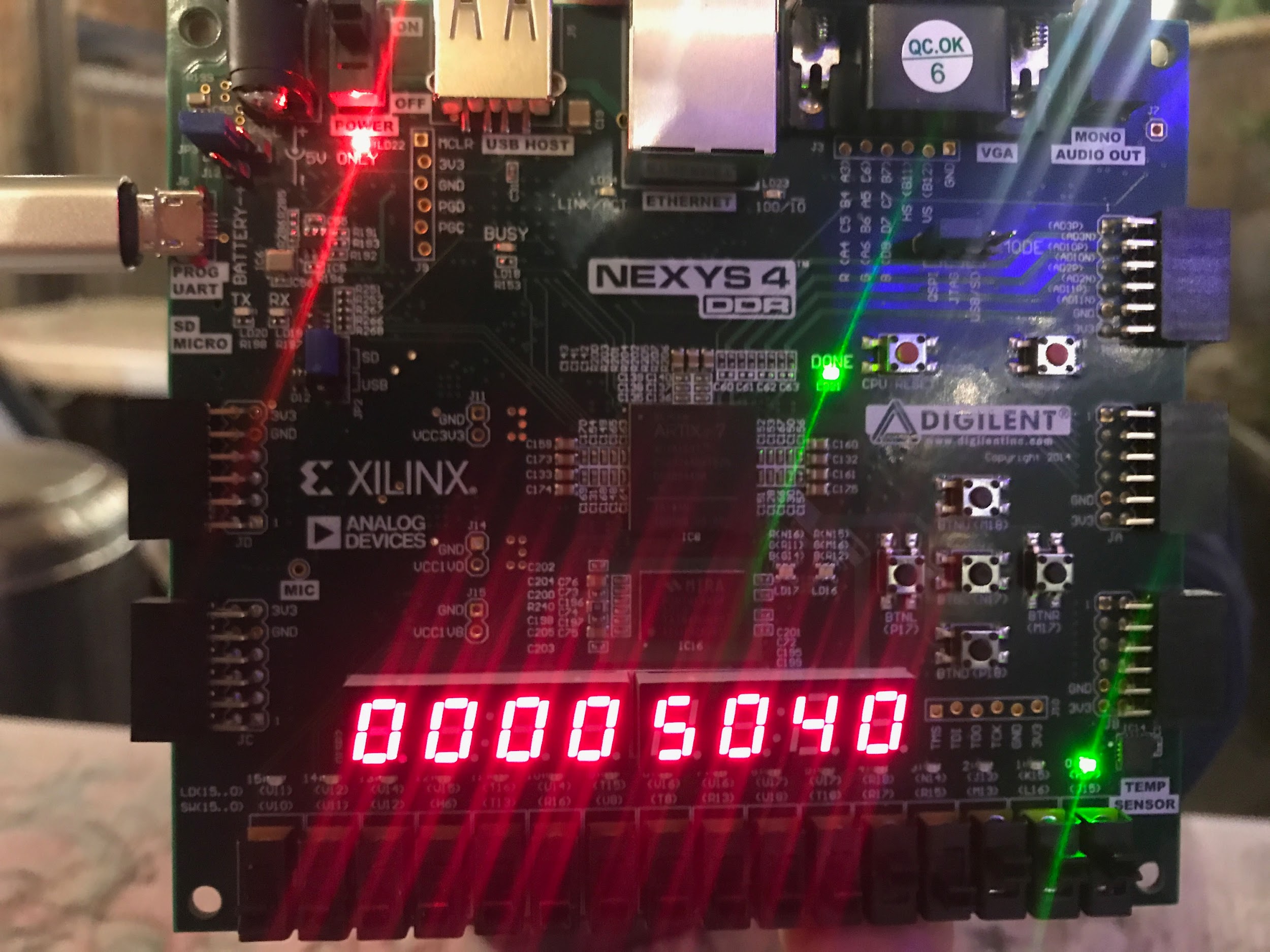
  
**Figure 3b:** datapath\_tb.v Tcl console output.

# FPGA Validation:

For hardware validation, shown in the pictures below of input 7 and o which started in state 0 with Go flag set. The results are shown in decimal with the output of 5040, which is correct for 7!, and output 1 which is correct for 0!.

****

**Figure 4a:** FPGA Validation Result of Factorial 0.

****

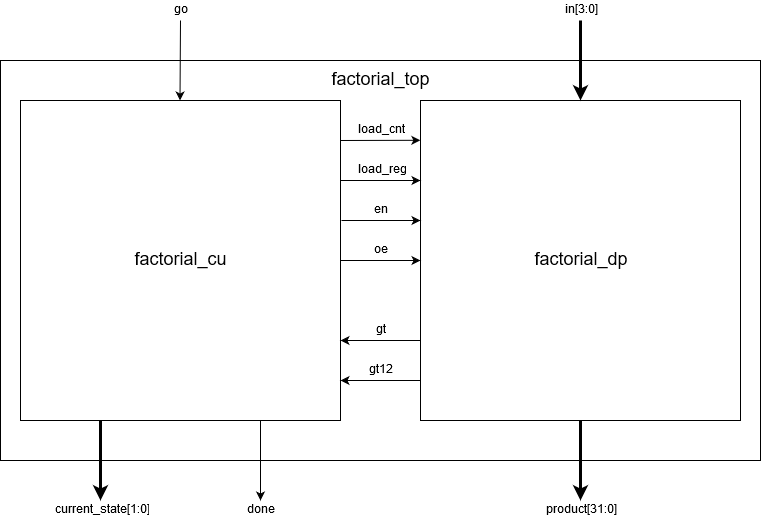
**Figure 4b:** FPGA Validation Result of Factorial 7.

# Conclusion:

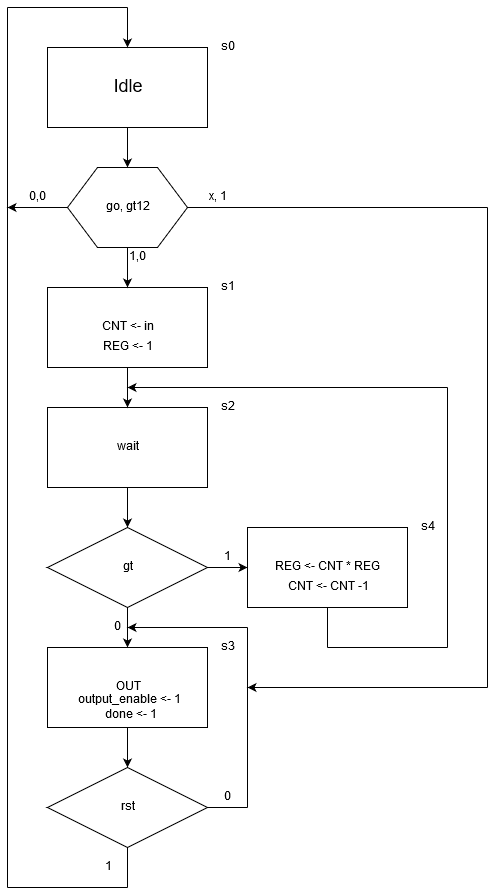
All tasks outlined in the *Procedure* section were successful. Conflict originally arose when implementing the top-level design on the FPGA board, but this was rectified and board functionality was completed as intended. Proper functionality of source code was also verified by simulations. This lab was a good test of retained knowledge from prerequisite courses.

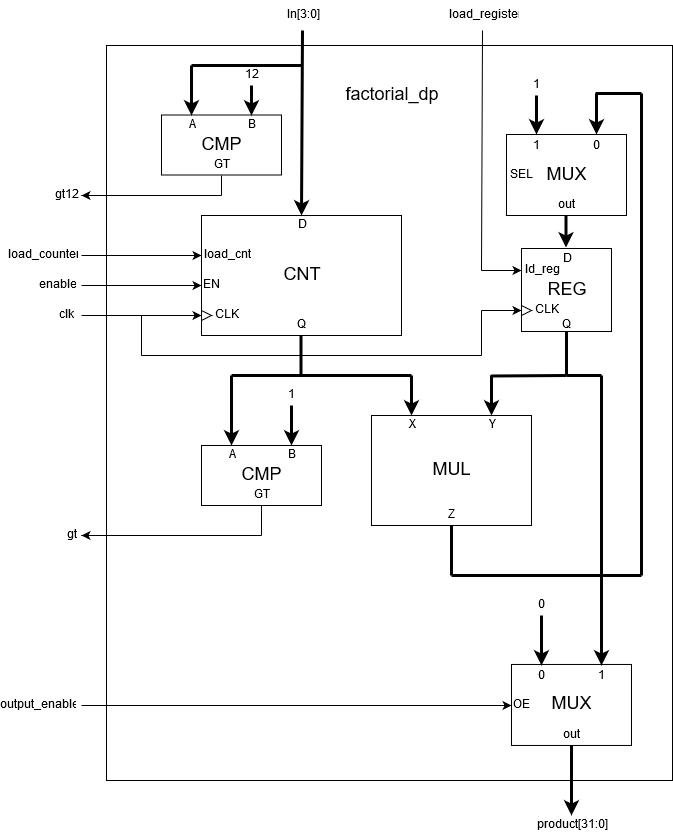
# Appendix:

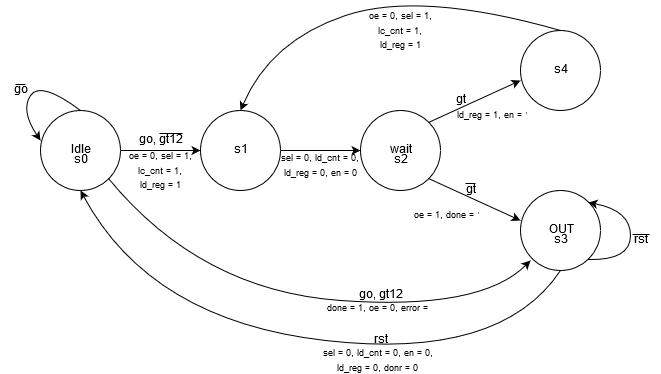
## Diagrams:

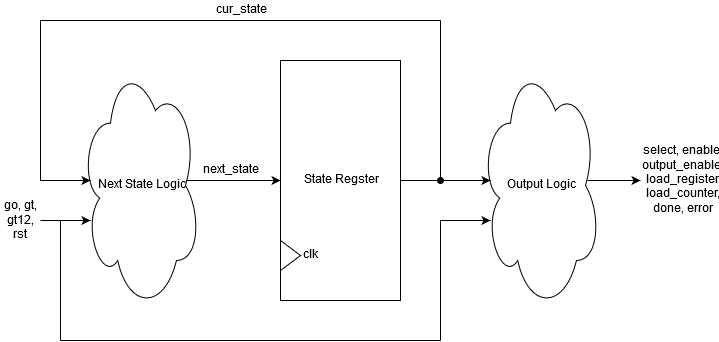


**Figure 5:** Top Module Block Diagram.

  
**Figure 6:** Control Unit Finite State Machine.

  
**Figure 7:** Data Path Module.

  
**Figure 8:** State Transition Diagram.

  
**Figure 9:** Next State Logic Diagram.

| CS | go | load\_cnt | load\_reg | cnt\_en | sel1 | sel2 | ud | GT | NS | DONE |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | x | x | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | x | x | 1 | 0 |
| 1 | x | 1 | 1 | 1 | 1 | 1 | x | x | 2 | 0 |
| 2 | x | 0 | 0 | 1 | 0 | 1 | x | 0 | 4 | 0 |
| 2 | x | 0 | 0 | 1 | 0 | 1 | x | 1 | 3 | 0 |
| 3 | x | 0 | 1 | 1 | 0 | 0 | 0 | x | 2 | 0 |
| 4 | x | 0 | 0 | 0 | 0 | 0 | x | x | 0 | 1 |

**Table 1:** State Output Table

## Source Code:

| factorial\_top.v |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  // Company:  // Engineer:  //  // Create Date:  // Design Name:  // Module Name: Factorial  // Project Name:  // Target Devices:  // Tool Versions:  // Description:  //  // Dependencies:  //  // Revision:  // Revision 0.01 - File Created  // Additional Comments:  //  //////////////////////////////////////////////////////////////////////////////////  module Factorial(input [3:0] n,  input Go, clk, rst,  output Done,  output [31:0] out);  wire [5:0] ctrl;  wire Is\_Gt;  Factorial\_CU U0 (.Go(Go), .clk(clk), .rst(rst), .Is\_Gt(Is\_Gt), .Done(Done), .ctrl(ctrl));  Factorial\_DP U1 ( .n(n), .ctrl(ctrl), .clk(clk), .rst(rst), .out(out), .Is\_GT(Is\_Gt));  endmodule |

| control\_unit.v |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  // Company:  // Engineer:  //  // Create Date:  // Design Name:  // Module Name: Factorial\_CU  // Project Name:  // Target Devices:  // Tool Versions:  // Description:  //  // Dependencies:  //  // Revision:  // Revision 0.01 - File Created  // Additional Comments:  //  //////////////////////////////////////////////////////////////////////////////////  module Factorial\_CU(input Go, clk, rst, Is\_Gt,  output reg Done,  output reg [5:0] ctrl);  reg [3:0] ns, cs;  parameter  IDLE = 6'b0\_0\_0\_0\_0\_0,  S1 = 6'b0\_1\_0\_1\_1\_0,  S2 = 6'b1\_0\_0\_0\_0\_0,  S3 = 6'b1\_0\_0\_0\_0\_1,  S4 = 6'b1\_1\_0\_0\_1\_0;    always @(Go, Is\_Gt, Done, ns, cs)  begin  case(cs)  4'd0:  begin  Done = 0;  if(Go) begin ns = 4'd1; end  else begin ns = 4'd0; end  end  4'd1: begin Done = 0; ns = 4'd2; end  4'd2:  begin  if(Is\_Gt) begin ns = 4'd4; Done = 0; end  else begin ns = 4'd3; Done = 0; end  end  4'd3:  begin  Done = 1;  ns = 4'd0;  end  4'd4:  begin  Done = 0;  ns = 4'd2;  end  default: ns = 4'd0;  endcase  end  always @(posedge clk, posedge rst)  begin  if(rst)  cs <= 0;  else  cs <= ns;  end  always @(cs, ctrl) //{Sel1, ce, ud, ld1, ld2, sel2} = ctrl;  begin  case(cs)  4'd0: ctrl = IDLE;  4'd1: ctrl = S1;  4'd2: ctrl = S2;  4'd3: ctrl = S3;  4'd4: ctrl = S4;  endcase  end  endmodule |

| datapath.v |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  // Company:  // Engineer:  //  // Create Date:  // Design Name:  // Module Name: Factorial\_DP  // Project Name:  // Target Devices:  // Tool Versions:  // Description:  //  // Dependencies:  //  // Revision:  // Revision 0.01 - File Created  // Additional Comments:  //  //////////////////////////////////////////////////////////////////////////////////  module Factorial\_DP(input [3:0] n,  input [5:0] ctrl,  input clk, rst,  output [31:0] out,  output Is\_GT);  wire [3:0] Cnt\_Reg;  wire [31:0] q\_out, aluout, q2\_out, mux\_out1, mux\_out2;  wire IS\_equal;  Ud\_Cnt\_4 u1(.D(n), .LD(ctrl[2]), .UD(ctrl[3]), .CE(ctrl[4]), .CLK(clk), .RST(rst), .Q(Cnt\_Reg));  comparater u2(.A({28'd0,Cnt\_Reg}), .B(32'd1), .greater(Is\_GT));  Dreg u3(.D(mux\_out1), .en(ctrl[1]), .clk(clk), .rst(rst), .Q(q\_out));  Alu u4(.in1(q\_out), .in2({28'd0,Cnt\_Reg}), .c(2'b00), .aluout(aluout));  Mux u6 (.in1(32'd1), .in2(aluout), .sel(ctrl[5]), .out(mux\_out1));  Mux u7 (.in1(32'd0), .in2(aluout), .sel(ctrl[0]), .out(out));  endmodule |

| factorial\_FPGA.v |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  // Company:  // Engineer:  //  // Create Date:  // Design Name:  // Module Name: Factorial\_fpga  // Project Name:  // Target Devices:  // Tool Versions:  // Description:  //  // Dependencies:  //  // Revision:  // Revision 0.01 - File Created  // Additional Comments:  //  //////////////////////////////////////////////////////////////////////////////////  module Factorial\_fpga(input [3:0] n,  input go, clk100MHz, control, rst,  output Done,  output [7:0] LEDOUT, LEDSEL );  supply1 [7:0] vcc;  wire DONT\_USE, clk\_5KHz;  wire [31:0] out;  wire [3:0] dig0, dig1, dig2, dig3, dig4, dig5, dig6, dig7;  wire [6:0] out0, out1, out2, out3, out4, out5, out6, out7;  Factorial U0 (.n(n), .Go(go), .clk(debouncedButton), .rst(rst), .Done(Done), .out(out));  bin2bcd32 U1 (.value(out), .dig0(dig0), .dig1(dig1), .dig2(dig2), .dig3(dig3), .dig4(dig4), .dig5(dig5), .dig6(dig6), .dig7(dig7));  showNumber U2 (.dig0(dig0), .dig1(dig1), .dig2(dig2), .dig3(dig3), .dig4(dig4), .dig5(dig5), .dig6(dig6), .dig7(dig7),  .out0(out0), .out1(out1), .out2(out2), .out3(out3), .out4(out4), .out5(out5), .out6(out6), .out7(out7));  led\_mux U3(clk\_5KHz, rst, {1'b1, out7}, {1'b1, out6}, {1'b1, out5}, {1'b1, out4}, {1'b1, out3}, {1'b1, out2}, {1'b1, out1},{1'b1, out0}, LEDOUT, LEDSEL);  clk\_gen U4(.clk100MHz(clk100MHz), .rst(rst), .clk\_4sec(DONT\_USE), .clk\_5KHz(clk\_5KHz));  button\_debouncer U5(.clk(clk\_5KHz), .button(control), .debounced\_button(debouncedButton));  endmodule |

| factorial\_FPGA.xdc |
| --- |
| set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { clk100MHz }];  create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports {clk100MHz}];  set\_property -dict { PACKAGE\_PIN J15 IOSTANDARD LVCMOS33 } [get\_ports { n[0] }];  set\_property -dict { PACKAGE\_PIN L16 IOSTANDARD LVCMOS33 } [get\_ports { n[1] }];  set\_property -dict { PACKAGE\_PIN M13 IOSTANDARD LVCMOS33 } [get\_ports { n[2] }];  set\_property -dict { PACKAGE\_PIN R15 IOSTANDARD LVCMOS33 } [get\_ports { n[3] }];  set\_property -dict { PACKAGE\_PIN H17 IOSTANDARD LVCMOS33 } [get\_ports { Done }];  set\_property -dict { PACKAGE\_PIN M17 IOSTANDARD LVCMOS33 } [get\_ports { rst }];  set\_property -dict { PACKAGE\_PIN N17 IOSTANDARD LVCMOS33 } [get\_ports { control }];  set\_property -dict { PACKAGE\_PIN R17 IOSTANDARD LVCMOS33 } [get\_ports { go }];  set\_property -dict { PACKAGE\_PIN K13 IOSTANDARD LVCMOS33 } [get\_ports { LEDOUT[0] }];  set\_property -dict { PACKAGE\_PIN K16 IOSTANDARD LVCMOS33 } [get\_ports { LEDOUT[1] }];  set\_property -dict { PACKAGE\_PIN P15 IOSTANDARD LVCMOS33 } [get\_ports { LEDOUT[2] }];  set\_property -dict { PACKAGE\_PIN L18 IOSTANDARD LVCMOS33 } [get\_ports { LEDOUT[3] }];  set\_property -dict { PACKAGE\_PIN R10 IOSTANDARD LVCMOS33 } [get\_ports { LEDOUT[4] }];  set\_property -dict { PACKAGE\_PIN T11 IOSTANDARD LVCMOS33 } [get\_ports { LEDOUT[5] }];  set\_property -dict { PACKAGE\_PIN T10 IOSTANDARD LVCMOS33 } [get\_ports { LEDOUT[6] }];  set\_property -dict { PACKAGE\_PIN H15 IOSTANDARD LVCMOS33 } [get\_ports { LEDOUT[7] }];  set\_property -dict { PACKAGE\_PIN J17 IOSTANDARD LVCMOS33 } [get\_ports { LEDSEL[0] }];  set\_property -dict { PACKAGE\_PIN J18 IOSTANDARD LVCMOS33 } [get\_ports { LEDSEL[1] }];  set\_property -dict { PACKAGE\_PIN T9 IOSTANDARD LVCMOS33 } [get\_ports { LEDSEL[2] }];  set\_property -dict { PACKAGE\_PIN J14 IOSTANDARD LVCMOS33 } [get\_ports { LEDSEL[3] }];  set\_property -dict { PACKAGE\_PIN P14 IOSTANDARD LVCMOS33 } [get\_ports { LEDSEL[4] }];  set\_property -dict { PACKAGE\_PIN T14 IOSTANDARD LVCMOS33 } [get\_ports { LEDSEL[5] }];  set\_property -dict { PACKAGE\_PIN K2 IOSTANDARD LVCMOS33 } [get\_ports { LEDSEL[6] }];  set\_property -dict { PACKAGE\_PIN U13 IOSTANDARD LVCMOS33 } [get\_ports { LEDSEL[7] }]; |

| datapath\_tb.v |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  // Company:  // Engineer:  //  // Create Date:  // Design Name:  // Module Name: dp\_tb  // Project Name:  // Target Devices:  // Tool Versions:  // Description:  //  // Dependencies:  //  // Revision:  // Revision 0.01 - File Created  // Additional Comments:  //  //////////////////////////////////////////////////////////////////////////////////  module dp\_tb();  reg [3:0] n\_tb;  reg [5:0] ctrl\_tb;  reg clk\_tb, rst\_tb;  wire [31:0] out\_tb;  wire Is\_GT\_tb;  integer i;    Factorial\_DP DUT( .n(n\_tb), .ctrl(ctrl\_tb), .clk(clk\_tb), .rst(rst\_tb), .out(out\_tb), .Is\_GT(Is\_GT\_tb));  parameter  IDLE = 6'b0\_0\_0\_0\_0\_0,  S1 = 6'b0\_1\_0\_1\_1\_0,  S2 = 6'b1\_0\_0\_0\_0\_0,  S3 = 6'b1\_0\_0\_0\_0\_1,  S4 = 6'b1\_1\_0\_0\_1\_0;  task tick; begin #5 clk\_tb = 1; #5 clk\_tb = 0; end endtask  initial  begin  clk\_tb = 0; rst\_tb = 0;      for(i = 0; i < 13; i = i + 1)  begin  n\_tb = i;  ctrl\_tb = IDLE; tick;  ctrl\_tb = S1; tick;  while(Is\_GT\_tb)  begin  ctrl\_tb = S2; tick;  ctrl\_tb = S4; tick;  end  ctrl\_tb = S3; tick;  case(i)  4'd0: begin if(out\_tb != 0) $display("Error with 0"); end  4'd1: begin if(out\_tb != 1) $display("Error with 1"); end  4'd2: begin if(out\_tb != 2) $display("Error with 2"); end  4'd3: begin if(out\_tb != 6) $display("Error with 3"); end  4'd4: begin if(out\_tb != 24) $display("Error with 4"); end  4'd5: begin if(out\_tb != 120) $display("Error with 5"); end  4'd6: begin if(out\_tb != 720) $display("Error with 6"); end  4'd7: begin if(out\_tb != 5040) $display("Error with 7"); end  4'd8: begin if(out\_tb != 40320) $display("Error with 8"); end  4'd9: begin if(out\_tb != 362880) $display("Error with 9"); end  4'd10: begin if(out\_tb != 3628800) $display("Error with 10"); end  endcase    end  $display("Sucess");  $finish;  end    endmodule |

| cu\_tb.v |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  // Company:  // Engineer:  //  // Create Date:  // Design Name:  // Module Name: cu\_tb  // Project Name:  // Target Devices:  // Tool Versions:  // Description:  //  // Dependencies:  //  // Revision:  // Revision 0.01 - File Created  // Additional Comments:  //  //////////////////////////////////////////////////////////////////////////////////  module cu\_tb();  reg Go\_tb, clk\_tb, rst\_tb, Is\_Gt\_tb;  wire Done\_tb;  wire [5:0] ctrl\_tb;  Factorial\_CU DUT(.Go(Go\_tb), .clk(clk\_tb), .rst(rst\_tb), .Is\_Gt(Is\_Gt\_tb), .Done(Done\_tb), .ctrl(ctrl\_tb));  parameter  IDLE = 6'b0\_0\_0\_0\_0\_0,  S1 = 6'b0\_1\_0\_1\_1\_0,  S2 = 6'b1\_0\_0\_0\_0\_0,  S3 = 6'b1\_0\_0\_0\_0\_1,  S4 = 6'b1\_1\_0\_0\_1\_0;    task tick; begin #5 clk\_tb = 1; #5 clk\_tb = 0; end endtask  initial  begin  clk\_tb = 0; rst\_tb = 0; Go\_tb = 1; Is\_Gt\_tb = 1;  tick;  if(ctrl\_tb != IDLE) $display("Error with IDLE");  tick;  if(ctrl\_tb != S1) $display("Error with S1");  tick;  if(ctrl\_tb != S2) $display("Error with S2");  tick;  if(ctrl\_tb != S4) $display("Error with S4");  tick;  Is\_Gt\_tb = 0;  tick;  if(ctrl\_tb != S3)  begin  $display("Error with S3");  if(Done\_tb != 1) $display("Error");  end  $display("Success");  $finish;  end  endmodule |

| factorial\_top\_tb.v |
| --- |
| `timescale 1ns / 1ps  //////////////////////////////////////////////////////////////////////////////////  // Company:  // Engineer:  //  // Create Date:  // Design Name:  // Module Name: Factorial\_tb  // Project Name:  // Target Devices:  // Tool Versions:  // Description:  //  // Dependencies:  //  // Revision:  // Revision 0.01 - File Created  // Additional Comments:  //  //////////////////////////////////////////////////////////////////////////////////  module Factorial\_tb();  reg [3:0] n\_tb;  reg Go\_tb, clk\_tb, rst\_tb;  wire Done\_tb;  wire [31:0] out\_tb;  integer i;  Factorial DUT (.n(n\_tb), .Go(Go\_tb), .clk(clk\_tb), .rst(rst\_tb),.Done(Done\_tb), .out(out\_tb));  task tick;  begin  #5 clk\_tb = 1;  #5 clk\_tb = 0;  end  endtask  initial  begin  n\_tb = 0; clk\_tb = 0; rst\_tb = 0;  Go\_tb = 0; tick; tick; tick; tick; tick; tick; tick; tick; tick; tick;  Go\_tb = 1;  for(i = 0; i < 11; i = i + 1)  begin  n\_tb = i;  tick;  while(!Done\_tb)  begin  tick;  end  case(i)  4'd0: begin if(out\_tb != 1) $display("Error with 0"); end  4'd1: begin if(out\_tb != 1) $display("Error with 1"); end  4'd2: begin if(out\_tb != 2) $display("Error with 2"); end  4'd3: begin if(out\_tb != 6) $display("Error with 3"); end  4'd4: begin if(out\_tb != 24) $display("Error with 4"); end  4'd5: begin if(out\_tb != 120) $display("Error with 5"); end  4'd6: begin if(out\_tb != 720) $display("Error with 6"); end  4'd7: begin if(out\_tb != 5040) $display("Error with 7"); end  4'd8: begin if(out\_tb != 40320) $display("Error with 8"); end  4'd9: begin if(out\_tb != 362880) $display("Error with 9"); end  4'd10: begin if(out\_tb != 3628800) $display("Error with 10"); end  endcase  end  $display("Success");  $finish;  end  endmodule |