Digital System Design ELEC373/473

Assignment Project Exam Help

https://tutorcs.com

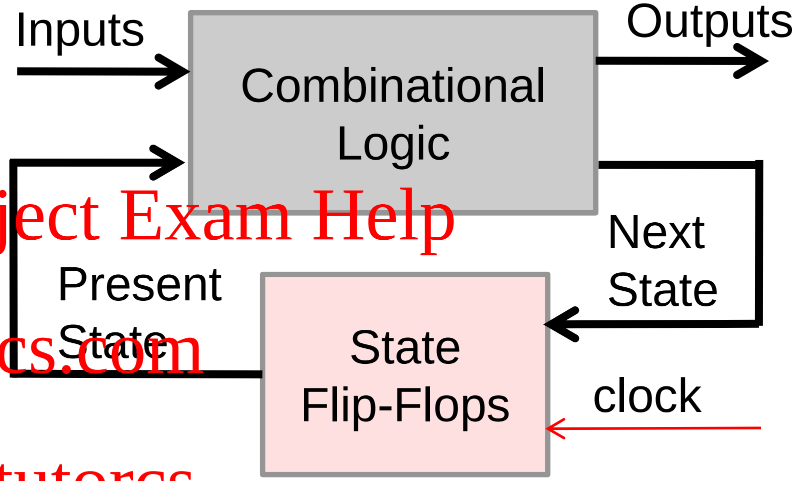
WeChat: cstutorcs

Algorithmic State Machine (ASM)

Coded In Verilog

State Machine Controller Structure

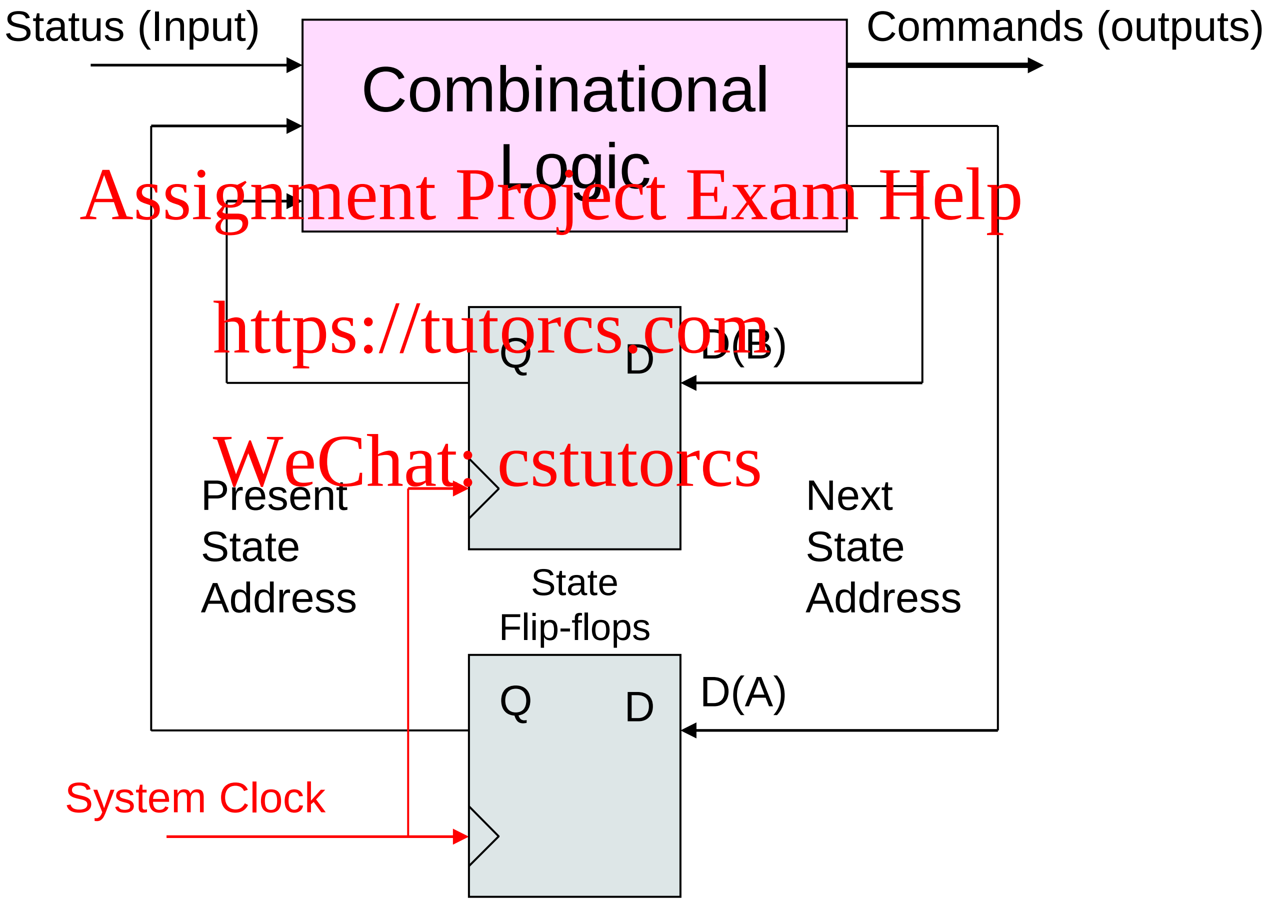
- All sequential circuits can be divided into a combinational block and a storage element block.



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

- There are two types of state machines:
  - Moore type outputs are a combinational function of only “Present State” signals.
  - Mealy type outputs are a combinational function of both “Present State” and “Input” signals.

Process Model



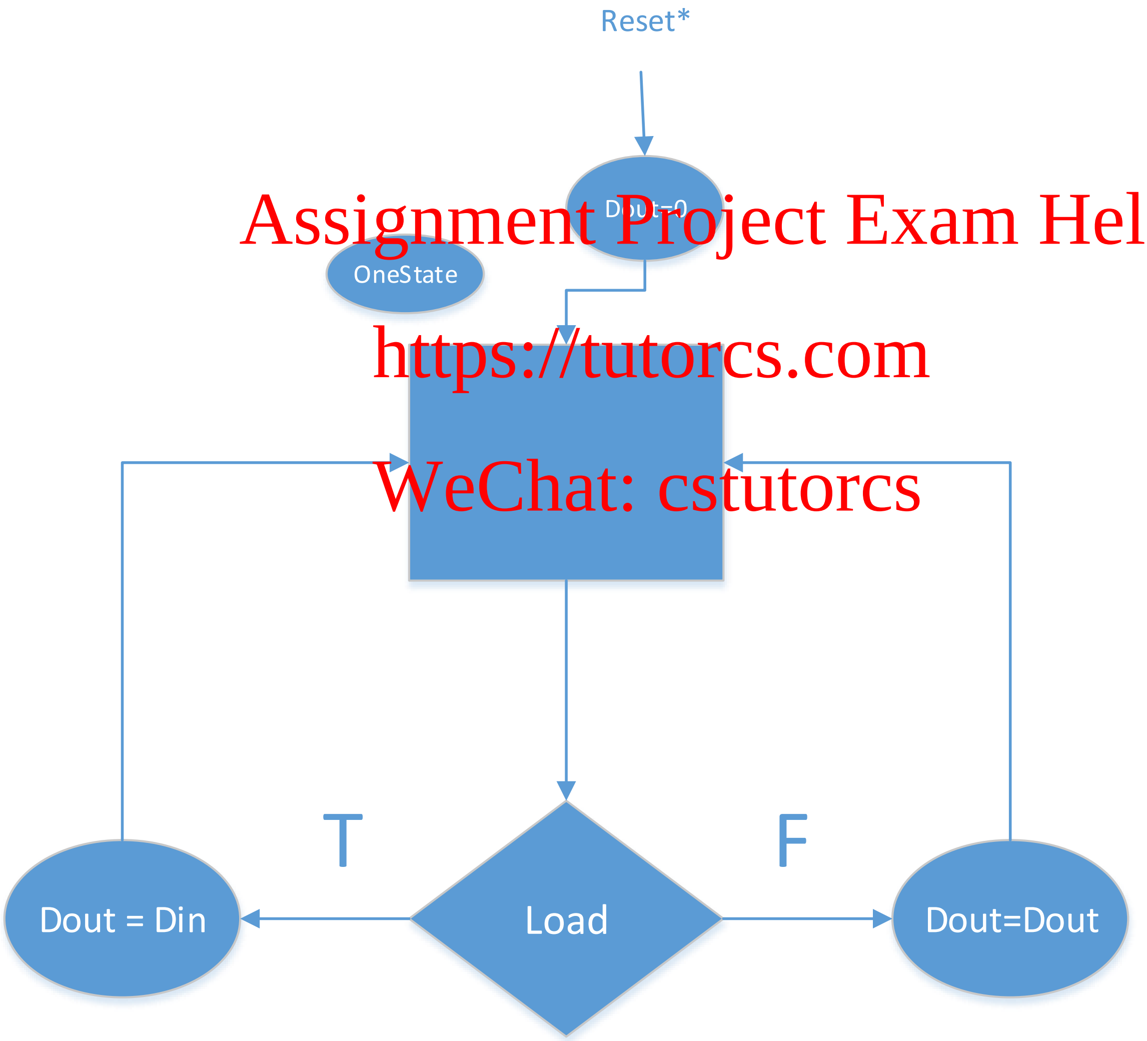
Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Verilog Sequential Template

```
module model_name (list of outputs and inputs);
external signal declarations
internal signal declarations
begin
          -- the state process defines the storage elements
          always @ (posedge (negedge) clock, optional reset)
          begin
                    verilog statements for storage elements (normally nonblocking)
          end

          -- the comb process defines the combinational logic
          always @ (level sensitivity list – usually includes all inputs and state vars)
          begin
                    verilog statements which specify combinational logic
                    (normally use a case statement to identify states)
          end
endmodule;
```

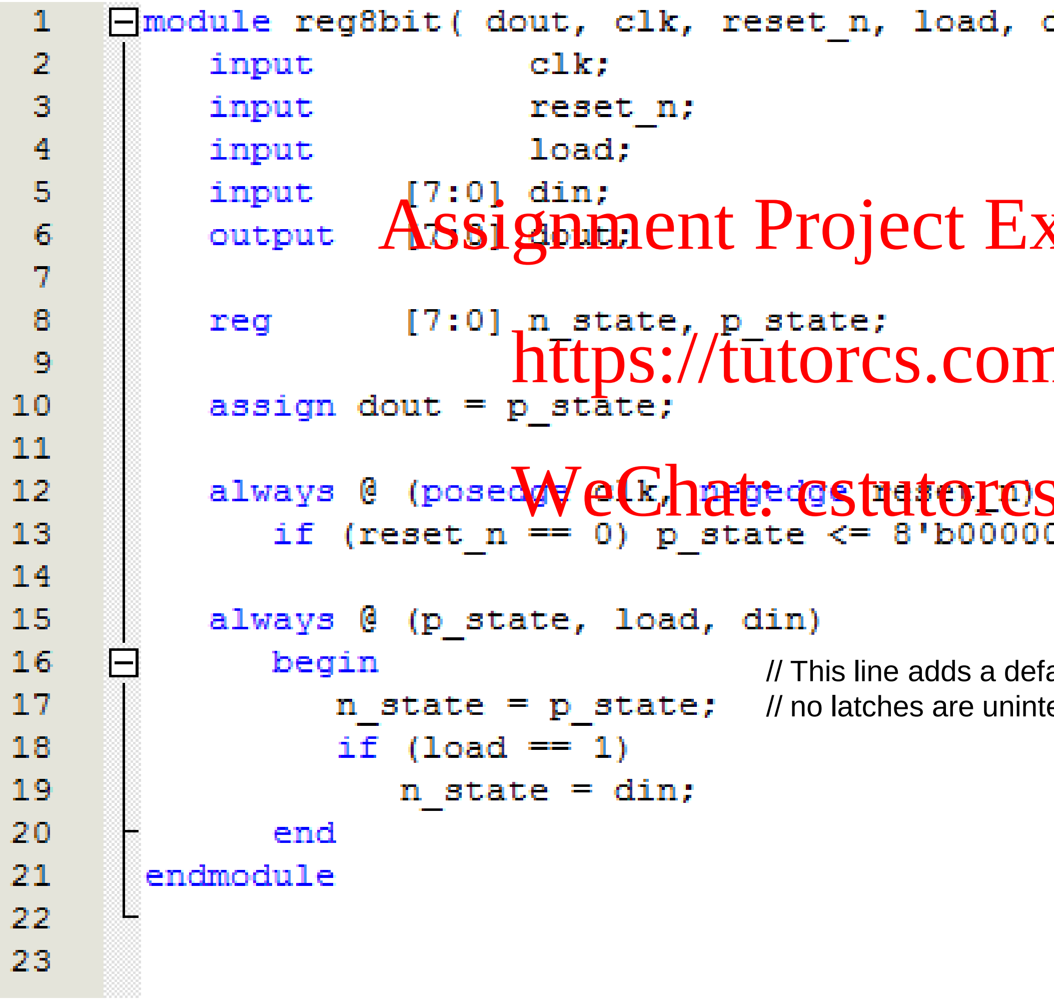
Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

8-bit loadable register with Asynchronous clear – ASM



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

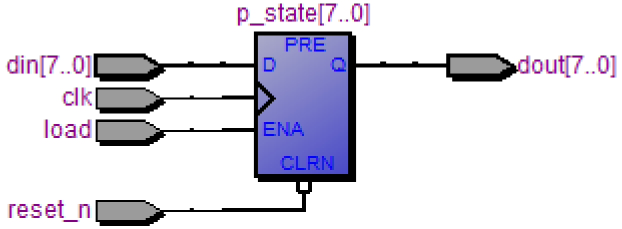
8-bit loadable register with Asynchronous clear



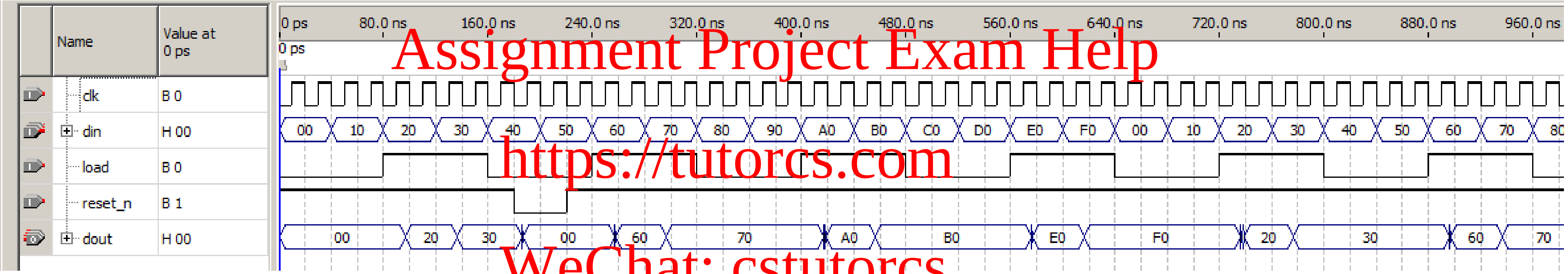
```
module reg8bit( dout, clk, reset_n, load, din);
   input          clk;
   input          reset_n;
   input          load;
   input    [7:0] din;
   output   [7:0] dout;

   reg      [7:0] n_state, p_state;

   assign dout = p_state;

   always @ (posedge clk, negedge reset_n)
      if (reset_n == 0) p_state <= 8'b00000000; else p_state <= n_state;

   always @ (p_state, load, din)
      begin
         n_state = p_state;
         if (load == 1)
            n_state = din;
      end
endmodule
```

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

// This line adds a default assignment so
// no latches are unintentionally synthesised



8-bit register Simulation

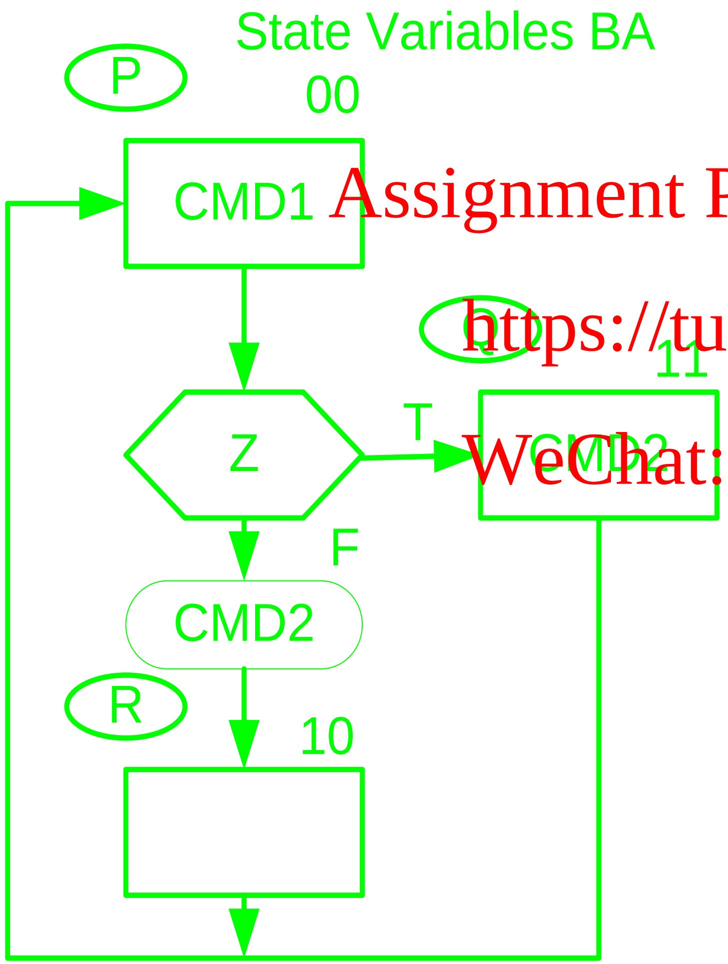


Name	Value at 0 ps
clk	B 0
din	H 00
load	B 0
reset_n	B 1
dout	H 00

0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800.0 ns
880.0 ns
960.0 ns
0 ps
00 10 20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0 00 10 20 30 40 50 60 70 80
00 20 30 00 60 70 A0 B0 E0 F0 20 30 60 70

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

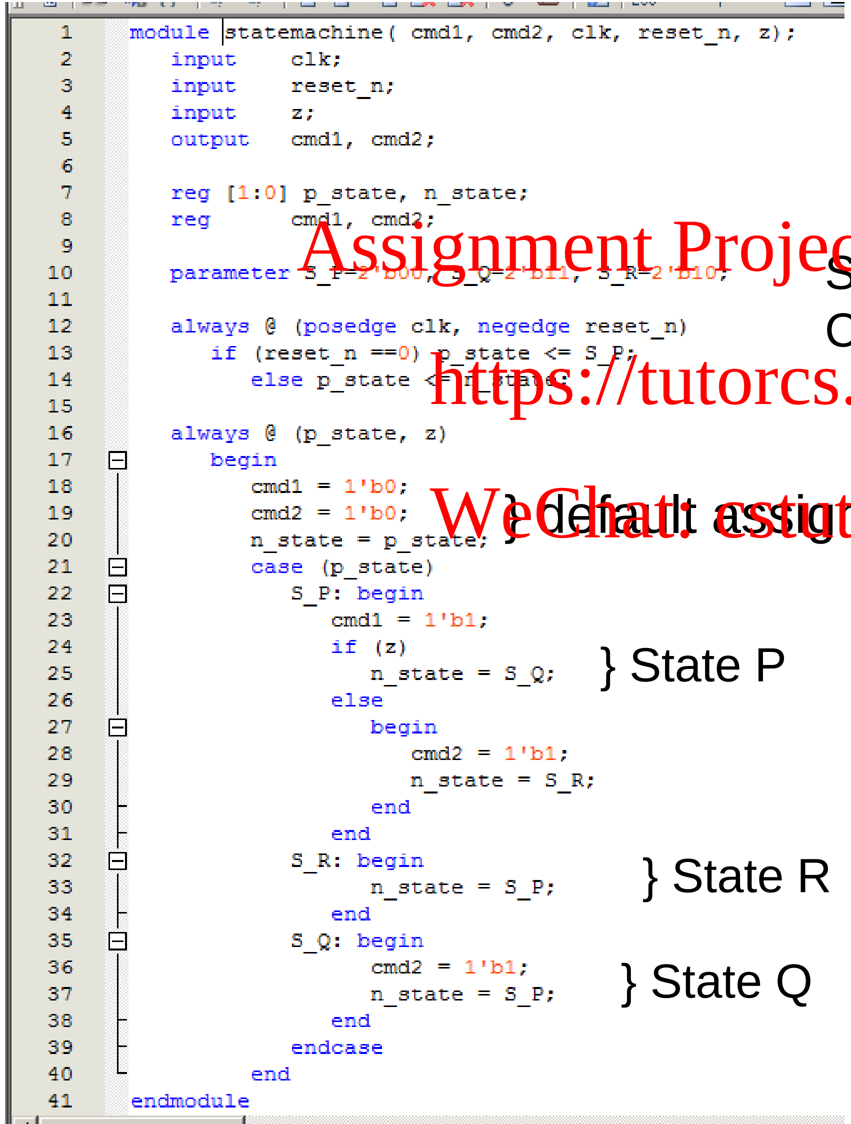
ASM Example 1



- 3 States (P,Q,R)
- Test input Z
- CMD1 Output is unconditional
- CMD2 Output is conditional and unconditional
- Also needs a reset, move to state P
- Mealy or Moore State Machine?

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Verilog Code for example 1



```verilog
module statemachine( cmd1, cmd2, clk, reset_n, z);
   input    clk;
   input    reset_n;
   input    z;
   output   cmd1, cmd2;

   reg [1:0] p_state, n_state;
   reg       cmd1, cmd2;

   parameter S_P=2'b00, S_Q=2'b11, S_R=2'b10;

   always @ (posedge clk, negedge reset_n)
      if (reset_n ==0) p_state <= S_P;
         else p_state <= n_state;

   always @ (p_state, z)
      begin
         cmd1 = 1'b0;
         cmd2 = 1'b0;
         n_state = p_state;
         case (p_state)
            S_P: begin
               cmd1 = 1'b1;
               if (z)
                  n_state = S_Q;
               else
                  begin
                     cmd2 = 1'b1;
                     n_state = S_R;
                  end
               end
            S_R: begin
                  n_state = S_P;
               end
            S_Q: begin
                  cmd2 = 1'b1;
                  n_state = S_P;
               end
            endcase
         end
endmodule
```

Split design into Sequential and Combinational Blocks

} default assignments to prevent latch synthesis

} State P

} State R

} State Q

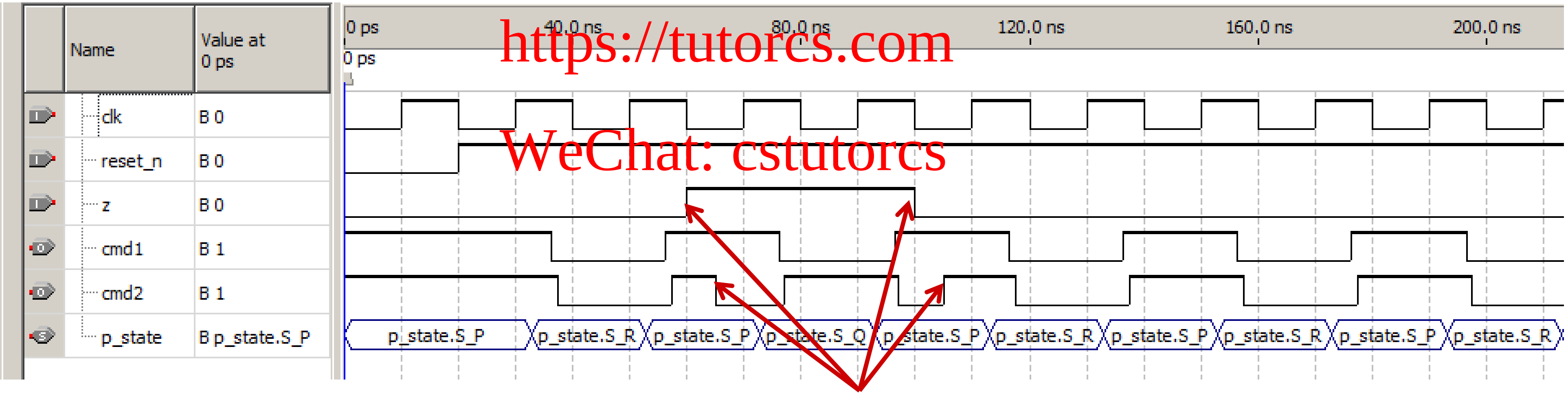
Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

Simulation of example 1

Assignment Project Exam Help



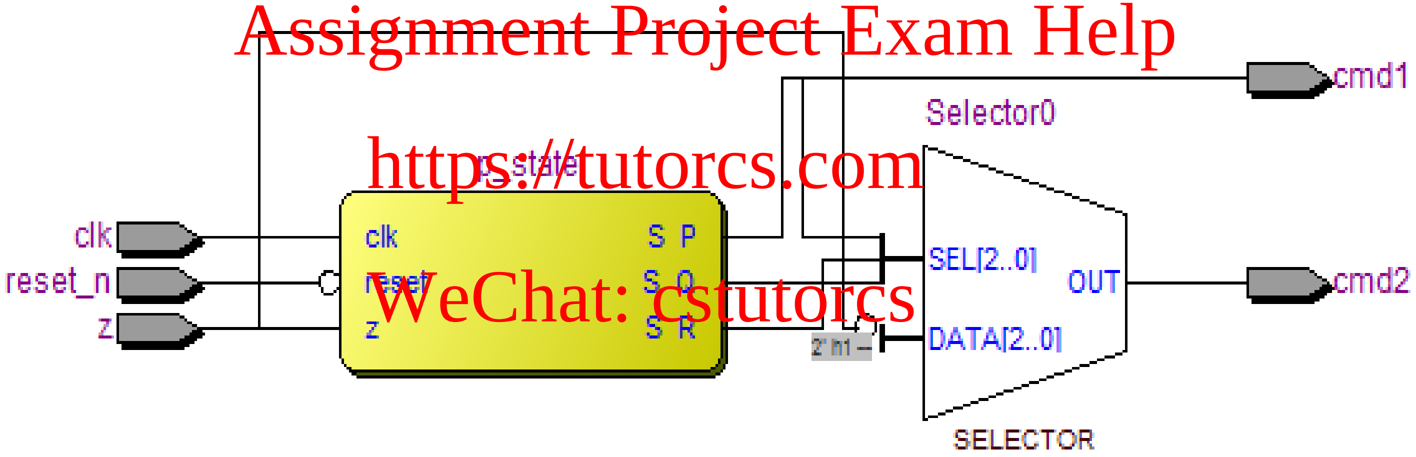
Name	Value at 0 ps
clk	B 0
reset_n	B 0
z	B 0
cmd1	B 1
cmd2	B 1
p_state	B p_state.S_P

https://tutorcs.com

WeChat: cstutorcs

Note: cmd2 changes
when input Z changes

RTL View of example 1

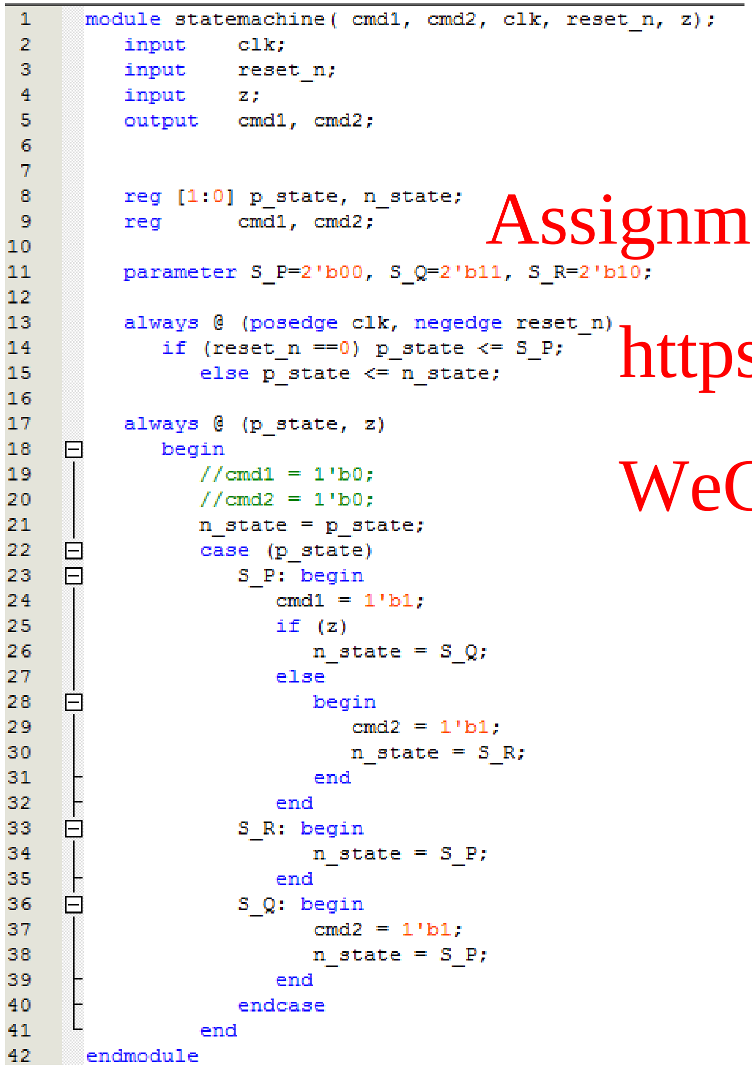


Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

Verilog Code without defaults

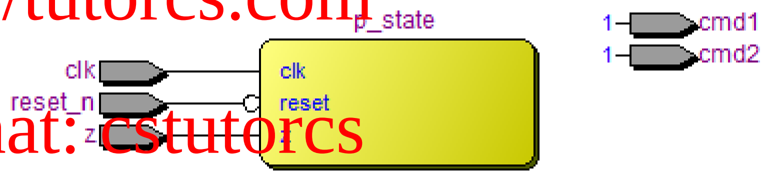


```verilog
module statemachine( cmd1, cmd2, clk, reset_n, z);
   input    clk;
   input    reset_n;
   input    z;
   output   cmd1, cmd2;


   reg [1:0] p_state, n_state;
   reg       cmd1, cmd2;

   parameter S_P=2'b00, S_Q=2'b11, S_R=2'b10;

   always @ (posedge clk, negedge reset_n)
      if (reset_n ==0) p_state <= S_P;
         else p_state <= n_state;

   always @ (p_state, z)
      begin
         //cmd1 = 1'b0;
         //cmd2 = 1'b0;
         n_state = p_state;
         case (p_state)
            S_P: begin
               cmd1 = 1'b1;
               if (z)
                  n_state = S_Q;
               else
                  begin
                     cmd2 = 1'b1;
                     n_state = S_R;
                  end
               end
            S_R: begin
                  n_state = S_P;
               end
            S_Q: begin
                  cmd2 = 1'b1;
                  n_state = S_P;
               end
            endcase
         end
endmodule
```

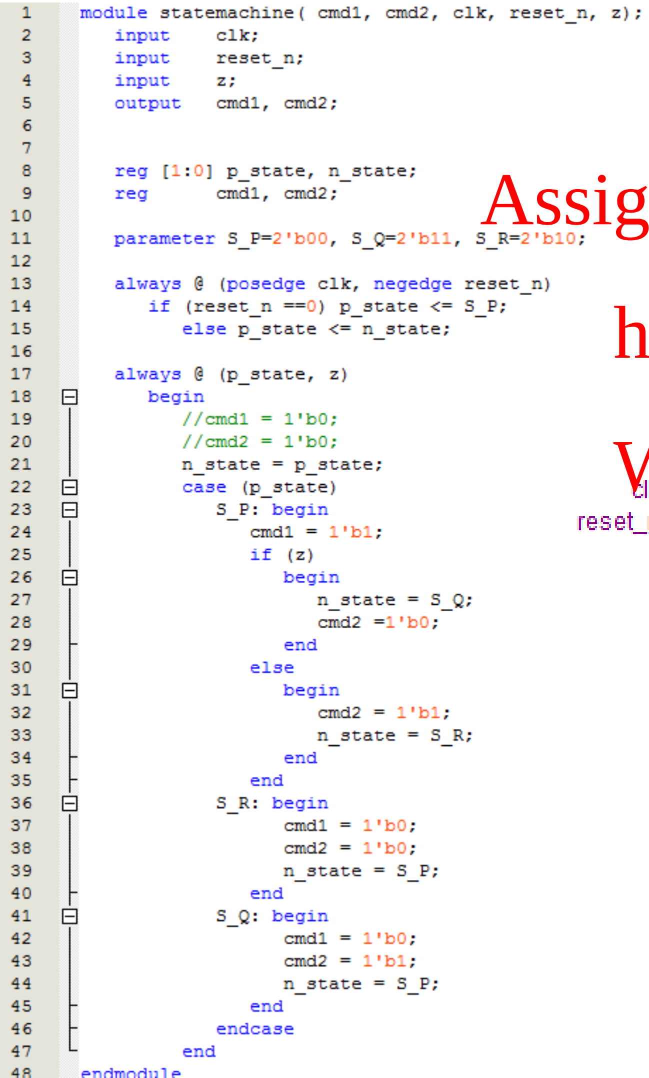
Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs



Verilog Code with ‘0’s added

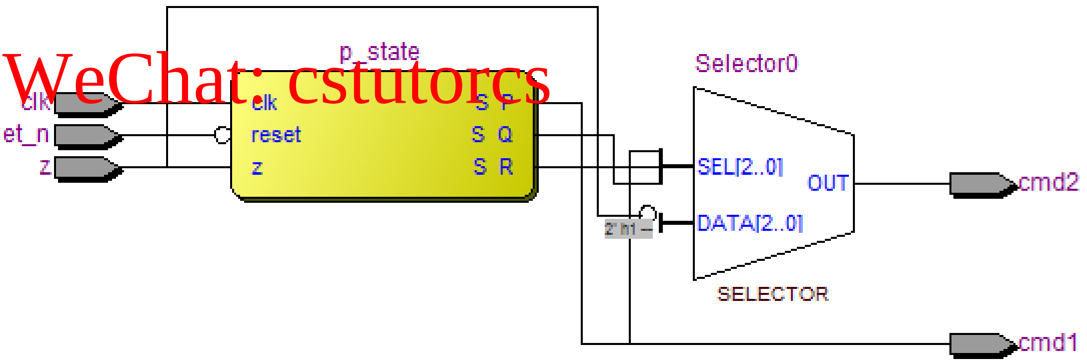


```
module statemachine( cmd1, cmd2, clk, reset_n, z);
   input    clk;
   input    reset_n;
   input    z;
   output   cmd1, cmd2;


   reg [1:0] p_state, n_state;
   reg      cmd1, cmd2;

   parameter S_P=2'b00, S_Q=2'b11, S_R=2'b10;

   always @ (posedge clk, negedge reset_n)
      if (reset_n ==0) p_state <= S_P;
         else p_state <= n_state;

   always @ (p_state, z)
      begin
         //cmd1 = 1'b0;
         //cmd2 = 1'b0;
         n_state = p_state;
         case (p_state)
            S_P: begin
               cmd1 = 1'b1;
               if (z)
                  begin
                     n_state = S_Q;
                     cmd2 =1'b0;
                  end
               else
                  begin
                     cmd2 = 1'b1;
                     n_state = S_R;
                  end
               end
            S_R: begin
                  cmd1 = 1'b0;
                  cmd2 = 1'b0;
                  n_state = S_P;
               end
            S_Q: begin
                  cmd1 = 1'b0;
                  cmd2 = 1'b1;
                  n_state = S_P;
               end
            endcase
         end
endmodule
```

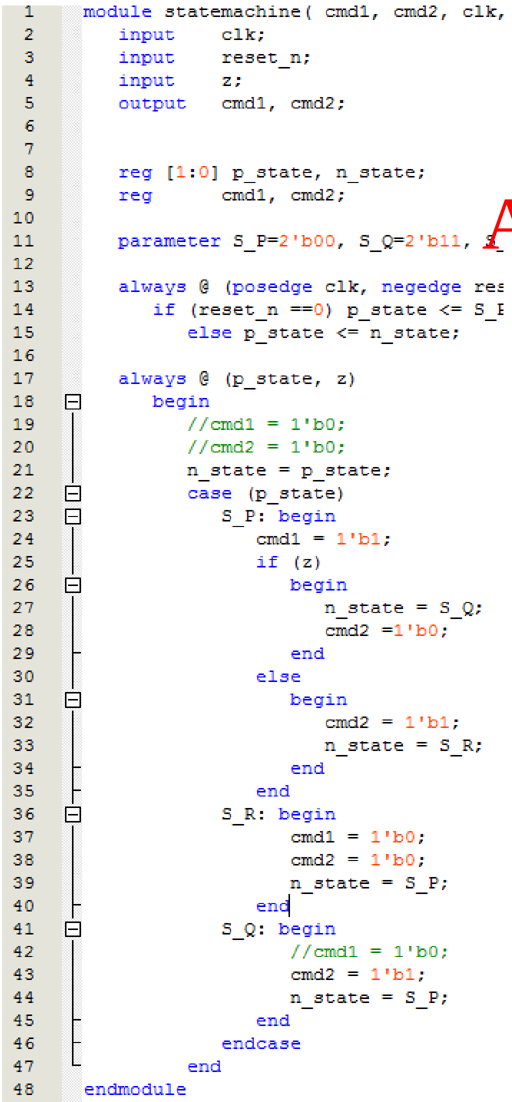
Assignment Project Exam Help

https://tutorcs.com

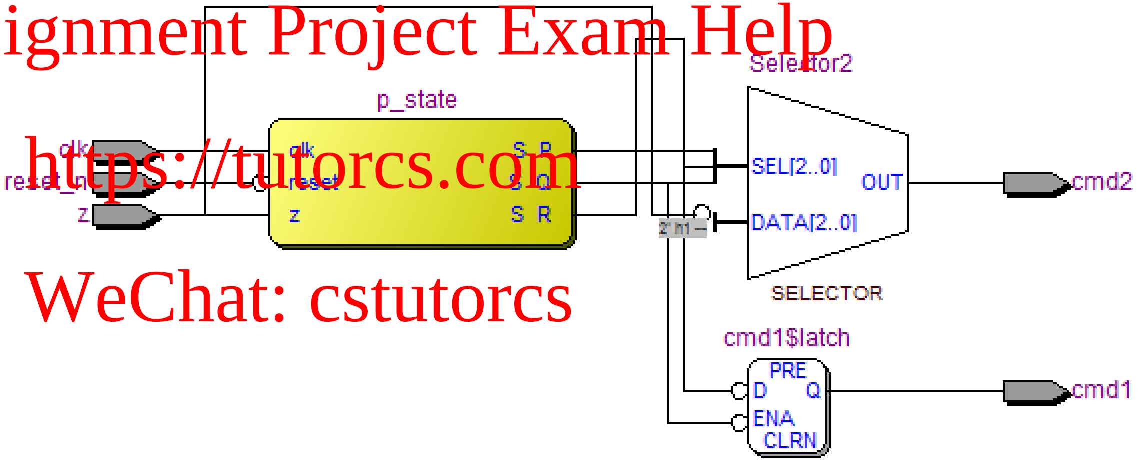
WeChat: cstutorcs



Verilog Code with an assignment missing



```
module statemachine( cmd1, cmd2, clk, reset_n, z);
   input    clk;
   input    reset_n;
   input    z;
   output   cmd1, cmd2;


   reg [1:0] p_state, n_state;
   reg       cmd1, cmd2;

   parameter S_P=2'b00, S_Q=2'b11, S_

   always @ (posedge clk, negedge res
      if (reset_n ==0) p_state <= S_P
         else p_state <= n_state;

   always @ (p_state, z)
      begin
         //cmd1 = 1'b0;
         //cmd2 = 1'b0;
         n_state = p_state;
         case (p_state)
            S_P: begin
               cmd1 = 1'b1;
               if (z)
                  begin
                     n_state = S_Q;
                     cmd2 =1'b0;
                  end
               else
                  begin
                     cmd2 = 1'b1;
                     n_state = S_R;
                  end
               end
            S_R: begin
                  cmd1 = 1'b0;
                  cmd2 = 1'b0;
                  n_state = S_P;
               end
            S_Q: begin
                  //cmd1 = 1'b0;
                  cmd2 = 1'b1;
                  n_state = S_P;
               end
            endcase
         end
endmodule
```



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

State flip-flop encoding

- With full encoding “n” flip-flops can encode 2^n states
  - 1 flip-flop can encode 2 states
  - 2 flip-flops can encode 4 states

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

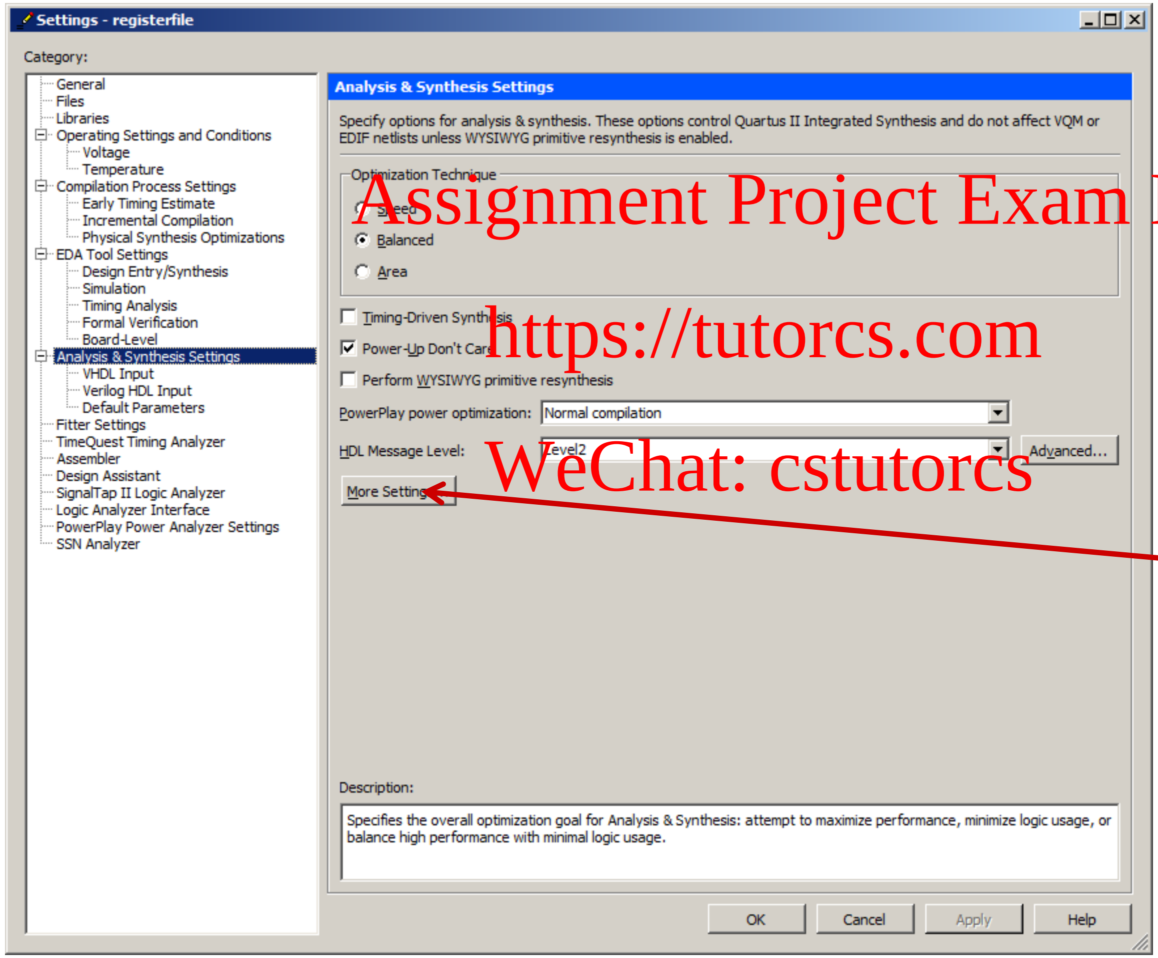
- With encoded state assignments, analysis has indicated that the maximum performance can be achieved if only one flip-flop changes its value between states i.e. a gray code implementation.
  - 00 ->01->11->10

One Hot encoding of ASMs

- One Hot encoding of ASMs uses one flip-flop per state.
  - Only one flip-flop is allowed 'on' at any time.
  - E.g. states are "00001", "00010", "00100", "01000" and "10000" for a five state ASM. All other states are illegal.
- One Hot encoding trades combinational logic for flip-flops.
  - Good for "flip-flop" rich implementation technologies.
  - Because the combinational logic is reduced, the length of the critical path can be reduced, resulting in a faster design. Speed increase is more significant for larger finite state machines.

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Quartus State Machine Encoding



Analysis & Synthesis Settings

Specify options for analysis & synthesis. These options control Quartus II Integrated Synthesis and do not affect VQM or EDIF netlists unless WYSIWYG primitive resynthesis is enabled.

Optimization Technique

○ Speed

● Balanced

○ Area

□ Timing-Driven Synthesis

☑ Power-Up Don't Care

□ Perform WYSIWYG primitive resynthesis

PowerPlay power optimization: Normal compilation

HDL Message Level: Level2

Description:

Specifies the overall optimization goal for Analysis & Synthesis: attempt to maximize performance, minimize logic usage, or balance high performance with minimal logic usage.

Assignment Project Exam Help

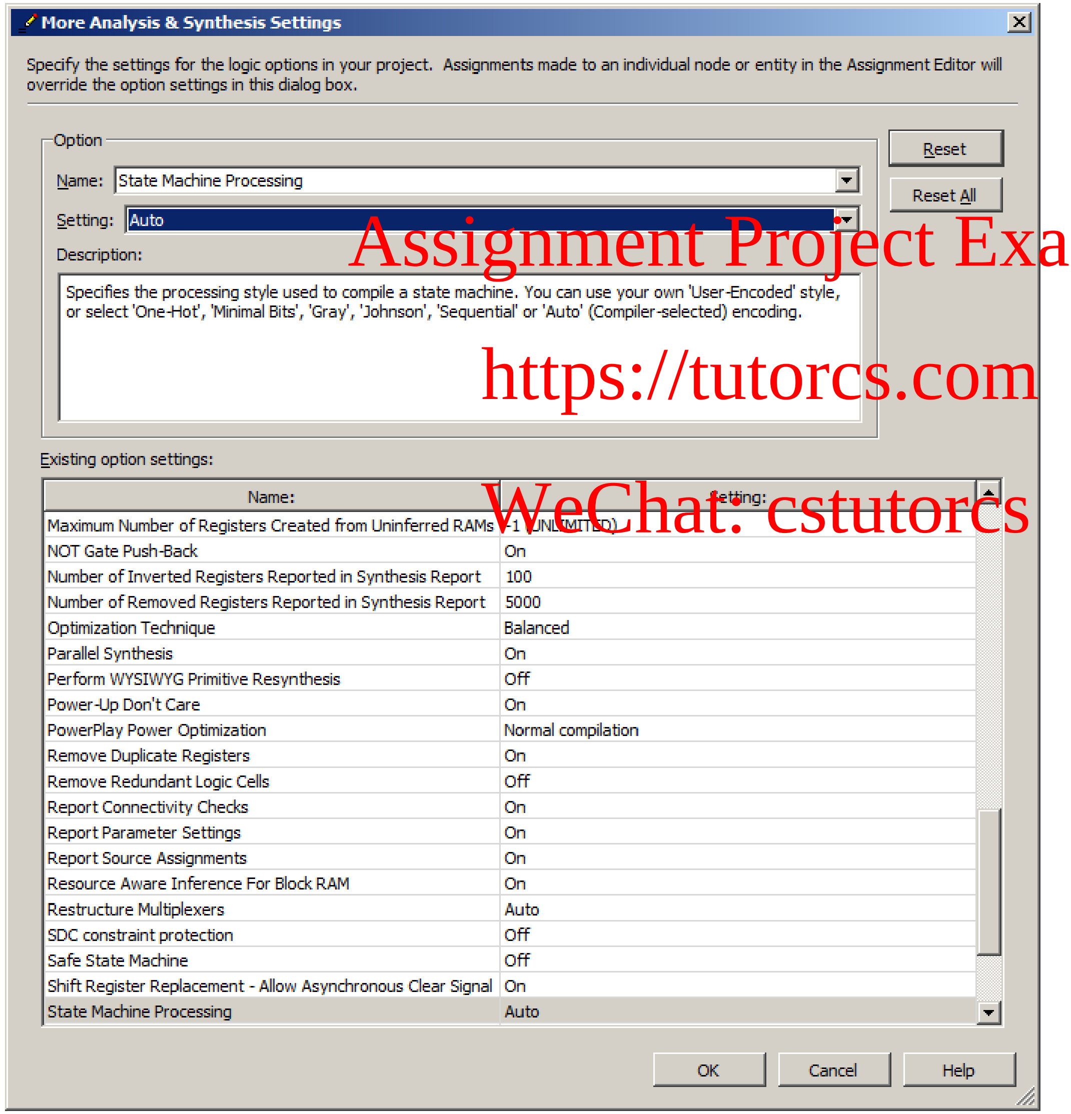
https://tutorcs.com

WeChat: cstutorcs

Assignments => Settings

More Settings

Quartus State Machine Encoding



Specify the settings for the logic options in your project. Assignments made to an individual node or entity in the Assignment Editor will override the option settings in this dialog box.

Option

Name: State Machine Processing

Setting: Auto

Description:

Specifies the processing style used to compile a state machine. You can use your own 'User-Encoded' style, or select 'One-Hot', 'Minimal Bits', 'Gray', 'Johnson', 'Sequential' or 'Auto' (Compiler-selected) encoding.

Existing option settings:

Name:	Setting:
Maximum Number of Registers Created from Uninferred RAMs	-1 (UNLIMITED)
NOT Gate Push-Back	On
Number of Inverted Registers Reported in Synthesis Report	100
Number of Removed Registers Reported in Synthesis Report	5000
Optimization Technique	Balanced
Parallel Synthesis	On
Perform WYSIWYG Primitive Resynthesis	Off
Power-Up Don't Care	On
PowerPlay Power Optimization	Normal compilation
Remove Duplicate Registers	On
Remove Redundant Logic Cells	Off
Report Connectivity Checks	On
Report Parameter Settings	On
Report Source Assignments	On
Resource Aware Inference For Block RAM	On
Restructure Multiplexers	Auto
SDC constraint protection	Off
Safe State Machine	Off
Shift Register Replacement - Allow Asynchronous Clear Signal	On
State Machine Processing	Auto

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs